# CREATIVE GAMES

## FOR THE TIMEX SINCLAIR 2068

ROBERT MAUNDER

# CREATIVE GAMES

## FOR THE TIMEX SINCLAIR 2068

ROBERT MAUNDER

CREATIVE COMPUTING PRESS
MORRIS PLAINS, NEW JERSEY

The Author and the Publisher have made every effort to verify the accuracy of the information contained in this book. However, neither the Author nor the Publisher assumes any responsibility for the use of this information, nor for any infringements of patents or other rights of third parties that may arise from the use of the information in this book. Neither the Author nor the Publisher assumes any liability for any damages that may result from the information contained herein.

Timex Sinclair 2068 is a registered trademark of the Timex Corporation.

Special thanks to David and James Grosjean for technical assistance.

The Working Timex Sinclair 2068
Copyright © 1984 by LINSAC.

All rights reserved. No portion of this book may be reproduced—mechanically, electronically, or by any other means including photocopying— without express written permission from the publisher.

Published by arrangement with LINSAC.

Library of Congress Cataloging in Publication Data

Maunder, Robert.
Creative games for the Timex/Sinclair 2068.

Rev. ed. of: The spectrum games companion. c1982
Includes index.
1. Timex Sinclair 2068 (Computer)—Programming.
2. Basic (Computer program language) 3. Computer games.
I. Maunder, Robert. Spectrum games companion. II. Title.
QA76.8.T494M38 1984 794.8′2 83-27264
ISBN 0-916688-24-0 (pbk.)

Creative Computing Press
39 East Hanover Avenue
Morris Plains, New Jersey 07950 USA

Manufactured in the United States of America

86 85 84 987654321

# CONTENTS

Page No.

CHAPTER ONE—STARTING ON TS2068 GAMES

1.1 Setting up the TS2068 1
1.2 Using This Book 5

CHAPTER TWO — DESIGNING TS2068 GAMES

2.1 Introduction 7
2.2 A Good Game 8
2.3 How to Design a Good Game 9

CHAPTER THREE—NUMBER GAMES

3.1 Crack the Combination 17
3.2 Money Match 20
3.3 Recognize the Enemy 25

CHAPTER FOUR—WORD GAMES

4.1 Awful Anagrams 31
4.2 Pace the Plank 35
4.3 Shrink 38

CHAPTER FIVE—BOARD GAMES

5.1 Copycat 47
5.2 What's the Code? 50
5.3 Solitaire 54

CHAPTER SIX—SIMULATION GAMES

6.1 Sink the Ship 59
6.2 Headquarters 61
6.3 Dead Ducks 66

CHAPTER SEVEN—DICE GAMES

7.1 Higher-Lower 71
7.2 Triple 76
7.3 Playing the Numbers 80

CHAPTER EIGHT—CARD GAMES

8.1 Inbetween 87
8.2 Concentration 93
8.3 Blackjack 98

CHAPTER NINE—GRID GAMES

9.1 Treasure Trove 105
9.2 Grab the Goblin 108
9.3 Pursuit 112

INDEX 121

KEYBOARD GUIDE 122

# PREFACE

*Creative Games For the Timex Sinclair 2068* is aimed at TS2068 owners who want to know and use their computer more fully through the medium of games, whether they are learning with the machine for the very first time or whether they are more experienced users. For beginners there is a simple guide to entering programs, and each program listing is clearly presented with accompanying instructions on playing the game. For those more advanced in programming skills, a complete program design method is introduced, each games program is explained and documented fully, and many programming techniques are included in the form of program notes. Games are organized into sections for ease of reference, e.g. number games, word games, card games and board games.

Even the most seasoned computer professional will admit to enjoying computer games, and the selection included in this book is made up both of old favorites, particularly if the program method can be carefully studied, and completely original games. Using this text together with the TS2068 manual, the reader can progress from just playing the games to understanding their structure, modifying them and eventually designing original creations. Machine code routines are not included in any program in order that further reference books need not be examined. This book aims to be a self-contained guide to advanced TS2068 Basic programming, while also providing ready-to-use games for new TS2068 owners.

It is recommended that the reader who wishes to initially use this text just as a source of games should read Chapter 1 and then commence with the program of their choice. Readers who have already programmed the TS2068 may wish to skip Chapter 1 and start with Chapter 2. Subsequently the chapters are relatively self-contained and can be considered in any order. The programs in a given chapter should be read through in sequence as material is introduced progressively.

# CHAPTER ONE

# STARTING ON TS2068 GAMES

This chapter is designed for readers who have recently acquired the TS2068 and who want to use it, at least initially, for playing games.

## 1.1 SETTING UP THE TS2068

### Make Sure it All Works

In spite of one's natural desire to start using the computer immediately it is very important to make sure that everything works properly before trying to type in a large game. To this end, the instructions in Chapter 1 of the TS2068 manual should be followed closely regarding plugging in all the bits and pieces. Initially, try and choose a place where you can do this on your own as a multitude of fingers poking around with leads and plugs often does not help.

Once the copyright line appears on the television screen you are on the right track, but it is still important to check two items before attempting the spectacular: the television must be tuned in precisely in order to fully appreciate the range of TS2068 colors, and also the cassette linkage must be tested. There is nothing more frustrating than spending an hour entering a program, only to find that it cannot be stored on cassette and is therefore going to be lost when the TS2068 is switched off.

To check the TV tuning we will run a short program to give a tuning picture, as with television test transmissions. However, before this can be done, the reader must be familiar with typing in programs.

### Using the Keyboard

The means by which information is entered into the TS2068's memory is generally by pressing keys on the keyboard. Unfortunately, it is not a simple case of looking at a program listing and typing it in letter by letter; this will not work. The TS2068 uses many special words (**keywords**) in programs which must be entered by pressing the appropriate keyword key. Sometimes the user may have to press several other keys before the TS2068 is ready to recognize the keyword. To gain an insight into this, the reader is directed to Chapter 2 of the TS2068 manual.

In order to type in a program it is necessary to appreciate a little theory. A TS2068 program is a list of numbered lines. Each line may have several instructions, separated by colons. At the end of a typed line the user must press ENTER. If the line does not move from the base of the TV screen then something is wrong; a flashing question mark at the end of the line indicates this.

To relieve the boredom of typing in a program the user may press RUN to get the TS2068 to obey the instructions entered so far. There is nothing wrong with this, providing one appreciates that a program will not work properly unless it is complete, and also the TS2068 screen may be left in a strange state, particularly as far as colors are concerned. More on this later.

Having mastered Chapter 2 of the TS2068 manual, one may start entering a program. In the last section of this book we have included a quick reference guide to keying in programs. This lists every symbol or word you will find in a program in this book and shows clearly how it is obtained. If you should find as you type in a program, that the sequence does not seem to work, then the TS2068 has not understood something typed in previously in that line. For example, if you were entering a line

```
PLOT x, 175: DRAW 0,—63
```

and you keyed ; instead of :, then the **DRAW** key (W in the keyboard guide) would just give you **W**. This is because the TS2068 thinks you are still somehow dealing with a PLOT instruction; it has not had the right separator (a colon) supplied. Such mistakes need to be corrected, as described in Chapter 2 of the TS2068 manual.

Whenever a TS2068 key is pressed, a faint click may be heard—this is not from the key itself but is emitted from the ''loud'' speaker inside the TS2068. This confirmatory sound may be made more audible by entering

```
POKE 23609,50 (and ENTER)
```

Any subsequent key depressions are heard as a beep. In fact you may like to experiment with values higher than 50 to see what you prefer best. Whatever value you use, the note will remain the same until you switch off the TS2068. There are one or two other general rules to be noted before starting to type in a program. Zero (0) and the letter O (O) are not the same and must be entered differently on the keyboard. Secondly, let us consider spaces. Generally spaces in program listings are provided by the TS2068 itself. For example key P in K mode gives

```
space PRINT space
```

The user can add further spaces (in fact these can add clarity to a listing—see Chapter 2) but these are irrelevant to the meaning of the program. The only place where spaces may be specifically needed in a program listing is between quotation marks e.g. in a PRINT statement. Failure to enter the proper number of spaces will mean that the program is not completely correct. It may work but certainly the screen layout will not be as intended. Also, if the symbol l is encountered, unless it is in a keyword (e.g. PRINT), it is always the lowercase letter l (as in lion).

One final hint. As stated above, when entering a listing the user may type RUN half way through to see what happens and this may result in strange things, particularly colors, on the screen which continue even when the listing returns. To overcome this, enter the following line before starting to key in a program.

```
9999 BORDER 7: PAPER 7: INK 0: CLS: STOP
```

Then if color problems are found, type

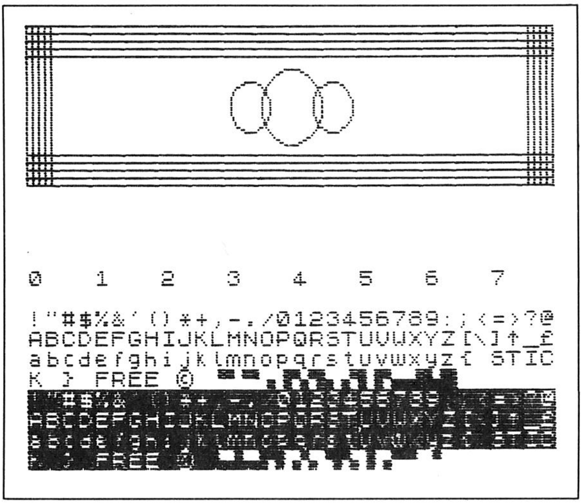
```
GO TO 9999
```

Line 9999 clears the screen and sets the screen colors to black characters on a white background, the same state as when the TS2068 is first switched on.

The reader may now like to practice using the keyboard with the **Keyboard Guide** found at the back of this manual. The TV tuning test program below is a good starting point.

## Adjusting the Television

If you are using a black and white set, tuning is much more straightforward. Often with a color set it is difficult to obtain both good colors and clear symbols. The test program listed below eventually sets up a screen with a pattern like this.



```
0    1    2    3    4    5    6    7

!"#$%&'()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_£
abcdefghijklmnopqrstuvwxyz{ STIC
K } FREE ©
```

The numbered blocks in the center are the colors, 0 to 7, i.e. black, blue, red, magenta, green, cyan, yellow and white. To get just this section in order to tune in your TV set for the right colors, enter these instructions:

```
   5 INK 0: PAPER 7: BORDER 7: C
LS
  10 FOR c=0 TO 7
  20 FOR i=9 TO 12
  30 PRINT BRIGHT 1; PAPER c;AT
i,c*4;"    "
  40 NEXT i
  50 PRINT BRIGHT 1; PAPER 8; IN
K 9;AT 12,c*4;c
  60 NEXT c
```

The following lines produce the normal and inverse character sets.

```
 100 PRINT AT 13,0
 110 FOR v=0 TO 1
 120 FOR h=33 TO 143: PRINT INVE
RSE v;CHR$ h;: NEXT h
 130 PRINT
 140 NEXT v
```

Finally the line and circle pattern at the top of the screen may be produced by entering

```
 200 LET yc=143
 210 CIRCLE 108,yc,10: CIRCLE 12
8,yc,15: CIRCLE 148,yc,10
 220 FOR x=0 TO 12 STEP 3
 230 PLOT x,175: DRAW 0,-63
```

```
240 PLOT x+243,175: DRAW 0,-63
250 NEXT x
260 FOR y=112 TO 124 STEP 3
270 PLOT 0,y: DRAW 255,0
280 PLOT 0,y+51: DRAW 255,0
290 NEXT y
```

### Checking the Cassette Recorder

Assuming the television is now tuned in as well as possible to the TS2068, it seems a good idea to keep this test program for future use. You may have allocated a spare button on your set exclusively for the TS2068, in which case there may be no need to use the program again. However, it is often true that other settings such as contrast or brightness need different positions for ordinary TV watching compared with TS2068 watching. Naturally such trivia will not bother most readers, who will doubtless have devoted one of their many color portables exclusively to TS2068 use. The fact remains that often we need to store programs on cassette tape, and the program that was entered in the previous section seems a good one with which to start.

Chapter 4 of the TS2068 manual is an excellent description of the use of cassette storage and should be read carefully. You may like to call the program TVTEST and then attempt to SAVE it and then VERIFY it using the details in the chapter. If VERIFY is successful you may, without a twinge of conscience, pull the plug out on your TS2068 and relax in the happy knowledge that you can reload TVTEST again whenever you wish, providing you can find the same cassette. On this matter it is well worth being thorough in labelling and cataloguing cassettes to avoid the panic of overwriting, losing or otherwise botching up programs which took precious time to key in.

## 1.2 USING THIS BOOK

### Playing the Games

Once you are convinced that the TS2068 is fully operational, cassette linkage and all, and you have had some practice on the keyboard, it is an appropriate time to start using some of the games in this manual. Each game comes with a great deal of accompanying information, but the only parts to concern the reader at this stage are the description, the program listing and any footnotes following the listing. Use the description to find out about the game and how to play it and then enter the program from the listing, paying attention to any footnotes. For simplicity, games in the manual do not contain a detailed list of instructions on how to play **in the program itself**. If these are required, simply distill the main points from the written description and then include them as PRINT instruction(s) with line number(s) preceding the first line number in the listing

e.g. (The ★ symbol is used here to indicate a space.)

```
1 PRINT "In★this★game★you★try★to★beat★the★
computer★at★estimating★the★value★of★PI★to★one★
hundred★decimal★places.★★If★you★win★you★get★
10★points.★★If★you★lose★the★keyboard★is★electrified.
★★Good★luck!"
```

You may find that words get split between lines—to overcome this simply add extra spaces before the split word to push it onto the next line. After these instructions you need a line such as

```
4 INPUT "Press★ENTER★to★start★the★game:";LINE
z$: CLS
```

This line gets the player to press the ENTER Key when the instructions have been read. Then the game will begin.

**Progressing Further**

After playing a game, the reader may wish to make modifications to it. Sometimes modifications are suggested in the description. This is an excellent way of learning about programming, and reference should be made to the method and variables sections of the program description to understand more fully what the program is doing. Refer also to the TS2068 manual to see the meaning and effect of instructions.

From here you may wish to move on to designing your own programs, and Chapter 2 of this book will equip you to do this.

# CHAPTER TWO

# DESIGNING TS2068 GAMES

This chapter is designed for readers familiar with programming.

## 2.1 INTRODUCTION

*In the days of our forebears, before the coming of the TS2068, when the strange rites of ZX80, ZX81, and TS1000 were practiced, our people were made to worship the god of Memory-Saving. All manner of great and wise artifacts were sacrificed to this god and even the sacred ancient tablets of Understanding and Structure were broken to appease the wrath of the Memory-Saving god. The god would only accept offerings of weird and mystical sayings which few could comprehend and these were spread abroad and the god said that it was good. But now we rejoice in the dawning of a new age where the bonds of ZX80, ZX81, and TS1000 will be broken asunder and we exult in the rule of the TS2068. Once again the land will resound to cries of ''Modular Programming!'' and ''Easily Modified!'' as we together renounce the black arts of Memory-Saving and return to the wise and true sayings of the ancients.*

Programs in this book are designed to reflect good programming practice and documentation. It is suggested that readers familiar with programming should choose chapters or games which interest them, perhaps key in a program to try it out and then consider it in more depth. Each program is described under the headings of

(i) description
(ii) method
(iii) screen layout
(iv) variables
(v) program listing
(vi) notes

The method is designed according to the principles outlined in this chapter and is cross referenced to the program listing by means of annotated line numbers. Study of the method section is the best way of understanding how the program works. Any notable techniques used in the program are explained in the notes section.

This chapter is concerned with how a game may be brought from the idea stage to be a clear, well-structured working program on the TS2068. Most of the material applies to programs other than games and in fact to programming on other machines as well.

## 2.2 A GOOD GAME

What makes a good game program? This question may be considered in two parts—firstly what is a good game, and secondly, what is a good program?

### Qualities of Games

Different players prefer different games but generally a game needs to be challenging (containing the right balance of luck and skill), novel, and complex enough to retain interest but simple enough to pick up quickly. The game should provide the player with feedback as to performance—at least whether the game was won or lost, and preferably a graded score. Some programmers ignore the playing potential of programs they design and treat the exercise as a means to show off their technical skill in designing algorithms. The end result is a technically brilliant program which is no fun as a game.

Similarly one should consider whether the game chosen is suitable for playing on a computer. Programs exist where the computer acts as the board and players enter their moves through the keyboard. While this may be satisfactory for two or three players, particularly if the very fact that it is a **computer** game holds some fascination for them, it could certainly become cumbersome with six players—unless of course you networked six TS2068s together with a projection television as the display!

### Qualities of Games Programs

Any program, game or otherwise, should aim to be correct, efficient, easily modified, and idiot proof. In addition, games programs, in order to fulfill the aims previously outlined, should make good use of graphics, color and sound, incorporating clear screen layouts and sensible interaction with the player.

We therefore require our games program to work properly, in spite of possible mistakes by the player, to work quickly, and to enable us to change its features easily.

As the reader will appreciate, none of these objectives are completely attainable, and some are not entirely complementary. However if the programmer aims at nothing, that is exactly what will be achieved—nothing. Many brilliant people have come up with excellent methods of designing programs to meet many of these aims and although they may have made their proposals for programmers working on multimillion dollar mainframe computers, their ideas are just as relevant for the TS2068 programmer. We now consider some of these ideas.

## 2.3 HOW TO DESIGN A GOOD GAME

### Why Bother?

If you have an idea for a TS2068 game why not just start typing it in? The answer is that you will end up, probably several days later, with a program that may work but that makes Rube Goldberg creations look like masterpieces of structural design; and heaven forbid that you should find an error in it or wish to modify it, as this will be nearly impossible.

In the good old days before microcomputers polluted our shores, programmers used terminals linked to mainframe computers which held strict records as to how much time the programmers spent on the machine. Online time was severely limited and the computer manager could thrust a complete list of all unsuccessful program runs under the programmer's nose and ask for explanations. This meant that the poor artisan actually had to ensure that any time on the computer was well-spent. Precious seconds could not be spent staring at the terminal thinking of which software fiddle to bolt-on next.

Apart from the odd time when you need to "see what happens if...", your time on the TS2068 should be spent either in enjoying using a program or simply keying in a program that you have previously prepared on paper. A builder does not build a house by shoving a few bricks together and then wondering what to do next: there is a carefully designed plan from which the builder works. If this were not the case, people would have no confidence in the completed buildings, and the same applies to our programs.

From the author's own experience, as far as writing programs is concerned, **it is quicker to do it properly**. Much time is saved by discovering errors on paper rather than when on the machine. This disciplined approach is not easy, particularly when you have a brilliant idea for a program that you are bursting to implement. You may like to imagine a large red sign on the TS2068 keyboard which reads **"DON'T TOUCH until you know exactly what you are going to do."**

## Crystallizing the Idea

Ideas for games can spring from many different sources, whether as variants of existing games or as original material. However the idea must be crystallized into a **description** of the game before any further work can be done.

The designer may ask a number of questions in order to assist in clarifying the idea.

1. What will the screen look like?
2. What is the player expected to enter?
3. How will the game start?
4. How will the game end?
5. How can the player do well?
6. In what way is skill required?

Once these questions have been answered, a description of the game should be written, consisting of about a paragraph of text, as in the programs in this manual. At the same time the screen format can be sketched in draft form.

To understand the process more fully, let us consider the first game in the book, Crack the Combination. The author had the idea of designing a game in which the player tries to break into a safe with a combination lock. Let us consider the six questions above.

1. The **screen**. It will display the dial of a combination lock with one number uppermost. There will be a game title and room (somewhere) for messages to the player.
2. **Player's entries.** The player has to type in the number to which the dial is to be turned.
3. The **game starts** with the combination turned to the highest number, which we choose to be 50. A code has been set, consisting of three numbers (again three is a personal choice).
4. The **game ends** with either the player guessing the code or running out of attempts. Restrict the number of allowed attempts to ten. If the player loses, we will create an alarm sound.
5. The **player can do well** by breaking the code in the least

number of attempts, in which case we had better show the player how many attempts they have made. No—for greater realism we will display SECURITY UNITS which start at 10 and decrease.

6. **Skill.** Burglars use their hearing to detect when the combination dial turns past the right number. Let us do a similar thing with this game—make a tone sound which increases up to the code number and then falls again. Perhaps include a "click" when the number is passed, as well.

The reader will notice that many decisions about the nature and format of the game are being made at this stage. Obviously such decisions must be based upon a practical knowledge of what the TS2068 can and cannot be programmed to do, and this only comes with practice. Some of the more nebulous ideas envisaged above become clearer as the game **description** is written as a brief paragraph and as the **screen format** is drawn (see Crack the Combination in Chapter 3).

### The Program Method

Since we want to produce a program which is itself just a list of steps, it seems sensible to design the program method as a list of steps. A perfectly viable program method is

1. Start the game
2. Play the game
3. Finish the game

but this is obviously insufficient detail with which to go to the TS2068 and know exactly what we are going to type in. We will follow a system known in the trade as **stepwise refinement** which starts with the method in very broad detail, as above, and gradually fills in more and more detail to each step and parts of steps until the Basic program is finally arrived at. At each stage of improvement or refinement the series of steps could properly be called a program method. In fact the method as listed in the programs in this manual is just a refinement stage chosen to be fairly readable in reasonably plain English. More or less detailed stages could just as well have been chosen. At each refinement the method must be checked for correctness, perhaps by feeding through sample values.

Following this methodology means that we will arrive at a well-structured program which is comprehensible and easily modified. As we fill in more detail we get close to TS2068 Basic and it is good to start using Basic variable names. These should be chosen to be meaningful, but not long-winded as they would then be a pain in the pinkies to type in.

Our second shot at the method above may look like this:

```
1.  Set up the dial picture and starting values
2.  Repeat for the three dial numbers
    a)  Computer chooses number
    b)  Player tries to guess it
    c)  Computer gives appropriate response
3.  Game ends
```

Notice that sub-steps are introduced—step 2, 'Play the Game' now consists of repetition of three steps, (a), (b) and (c). The program designer now goes into further detail. For clarity we will keep the sub-step labels the same, e.g., the player's guess will always be 2(b).

It is quite possible to refine a single step right to the level of Basic if desired, providing the steps are self contained or modular. This may not however be a good approach in one's first attempt at program design, as the refinement of one step may reveal a logic error in the design of others. We are therefore producing the following:

```
1.  Initialize screen and variables
    a)  Draw dial and headings
    b)  Set security units (su) to 10
2.  Repeat for section counter c from 1 to 3
    a)  Choose random code number (n)
    b)  Entry of player's guess
        (i)   Enter dial position required (d)
        (ii)  Check whether valid
    c)  Response to guess
        (i)   Move dial, with sound, to value d
        (ii)  If guess (d)=code number (n), respond and go to
              step (d)
        (iii) Otherwise decrease security units (su) by 1 and
              respond
    d)  Repeat to step (2)
3.  End of game
    a)  If all three numbers found, congratulate and stop
    b)  Otherwise sound alarm and stop
```

As we start to use data names such as su, c and n it is best to start to compile a list of names used, with their meanings. This is in fact how the variables section of the program documentation is produced. Such a list ensures that a data name is not used twice unintentionally and also that the designer can quickly refer back to it, rather than having to remember names and their meanings.

To meet some of the "idiot-proofing" aims, a validation step is included at step 2(b) (ii). The level of validation required depends upon your assessment of the level of idiocy of the potential players. For brevity in most of the programs in this manual we have assumed that if the player is asked for a number then one will be entered, albeit possibly too large or small or of the wrong sign. This philosophy applies in the final program method for Crack the Combination as shown in Chapter 3 and in the program listing. If however the program designer wishes to be more thorough and is prepared to type in extra code, a better system is as follows: for every input always use a string variable, and then convert it to a number if required. Here is a suitable method to check for unsigned integers.

```
(i)    Enter value z$
(ii)   Remove any leading and trailing spaces of z$
(iii)  For counter i from 1 to length of z$
       a)  If character i of z$ is not a digit 0-9 then
           1)  Set valid = 0
           2)  Repeat to (iii)
(iv)   Set valid = 1
(v)    Exit
```

A much more sophisticated routine is needed to check for positive or negative numbers : the reader may like to design one and then ensure it is right by trying oddities such as 12  34, −20. and 3.9.6−4.

As previously stated the final "plain English" version of the method for Crack the Combination is shown in Chapter 3. In writing down these methods it is helpful to use indentation as shown, for greater clarity. The line numbers shown in the method in Chapter 3 would not be present at this stage—these are added when the working program is ready, in order to provide easy cross-referencing.

### Translating into Basic

While planning and refining the program method the designer is already thinking of how it will be done in Basic and is working on identifying sections which occur more than once with a view to making them closed subroutines (using GO SUB).

Once the final method (in English or perhaps pidgin-Basic) has been written down, we simply have to carry out a translation exercise, converting the steps into TS2068 Basic instructions. This stage is often called **coding** the program i.e. putting the instructions into a code—Basic.

This should not be a particularly creative task as the thinking has been done already. Do not be afraid to use sections from programs you have previously written in your new program—this can save a lot of time. Many program modules from this manual can be similarly borrowed.

The addition of REM lines to mark sections of the program corresponding to the method steps is useful. Another hint for increasing readability is to use extra spaces to indent Basic lines within a loop or following an IF statement. Neither of these techniques are used in this book since the overriding aim is ease of entering programs especially for beginners, but compare the section of code below with its equivalent in Crack the Combination in Chapter 3.

```
  49 REM ***Start of dial
             section loop
  50 FOR c=1 TO 3
  60  LET n=INT (RND*46+5)
  69  REM set BEEP tones
  70  FOR i=1 TO 50
  72  LET b(i)=25-(i<n)*(n-i)/2-
(i>n)*(i-n)/2
  74  NEXT i
  75  LET b(n)=-10
  79  REM Enter & check dial
          number
  80  INPUT FLASH 1;"Enter next
number:";d
  90  LET d=INT d: IF d<1 OR d>5
0 OR d=x THEN PRINT INK 2; FLASH
 1;AT 10,7;"UNAUTHORIZED ENTRY":
 GO TO 210
```

## Getting it Right

Helpful hint: when first coding the program add plenty of extra print statements to show what is happening. These may be removed when the program is working.

In Combination, for example, print out the random code numbers, have a print statement to show whenever a new program section starts, print the values of the BEEP tones, and so on. Such extra information serves to either confirm that a program is working or to show why it is not. It is not a vainly optimistic hope to suggest that your program will work the first time, providing you have designed it properly. Do people gaze in wonder and astonishment at architects whose buildings do not fall over when built? Of course not, and neither should you be amazed if our programs do not fall over.

Certainly some adjustments may need to be made, particularly if the programmer is new to TS2068 Basic, and at this stage who isn't? However, using built-in error diagnostics and extra print statements, either in the program or as direct commands outside the program, problems should soon be resolved.

However we are wise to remember that even this final stage of the stepwise refinement process needs to be checked thoroughly, even if the program seems to work. Choose and feed in a good variety of test values to the program to make sure that as many routes as possible are tried and tested and not found wanting.

When you are happy with the finished product, you should gather together all the pieces of paper produced in the design process, correct any inconsistencies (e.g. slight modifications that you have made to the method at a later stage), cross reference the method to the program listing (as shown in this book), and file this information for further reference. In this way, writing up a program is not a tiresome drawn-out exercise, but simply a brief collating of information already produced.

# CHAPTER THREE

# NUMBER GAMES

## 3.1 CRACK THE COMBINATION

### DESCRIPTION

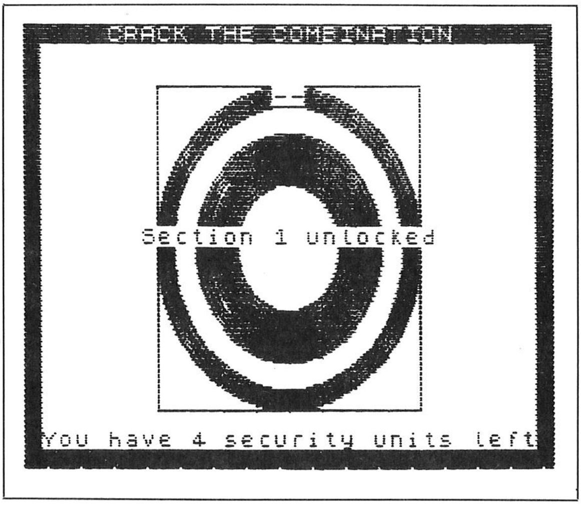
One of the first games that budding programmers generally put their hands to is Guess the Number. Here the computer chooses a number at random and the player has to find it in ten attempts, aided by hints such as 'TOO HIGH' from the computer. 'Crack the Combination' is of the same ilk, but is a bit more sophisticated.

An electronic combination lock to a safe has three code numbers between 1 and 50. The potential safebreaker has ten attempts to get all three numbers. The number chosen is keyed in and the dial turns to this number, emitting a tone, the pitch of which varies according to the closeness to the code number. As it gets closer to the number, the note rises, and as the number is passed, there is a buzz, and then the note falls again. If the player makes a mistake in entering the number then alarms are triggered. This also happens if security units run out (i.e. the ten attempts allowed). The alarm rings for quite some time!

### METHOD

```
1. Initialize screen and variables
   a) Draw dial and headings                                (5-45)
   b) Set dial number exposed (x) to 50                       (10)
   c) Set security units (su) to 10                           (10)
2. Repeat for section counter c from 1 to 3                   (50)
   a) Choose random code n                                    (60)
      (i) Set up corresponding array of BEEP tones (b)     (70-75)
   b) Entry of player's guess
      (i) Enter dial position required (d)                    (80)
      (ii) If invalid then                                    (90)
           A) Display message
           B) Set security units (su) to zero
           C) Go to step 3
      c)  Response to guess
          (i)  Move dial, with sound, to value d                  (100-120)
          (ii) If guess (d) = code number (n) then                   (130)
               A) Sound and message confirmation                  (140-150)
               B) Go to step d                                       (150)
          (iii) Otherwise
               A) Reduce security units (su) by 1 and display        (160)
               B) If su = 0 then                                     (170)
                    (I)  Display message
                    (II) Go to step 3
               C) If su = 1 then display warning message             (180)
               D) Go back to step 6                                  (190)
      d)  Repeat to step (2)                                         (200)
  3.  End of game
      a)  If units (su) = 0 then sound alarm                         (210)
      b)  Otherwise display message of congratulations               (220)
      c)  Stop
```

SCREEN LAYOUT



CRACK THE COMBINATION

Section 1 unlocked

You have 4 security units left

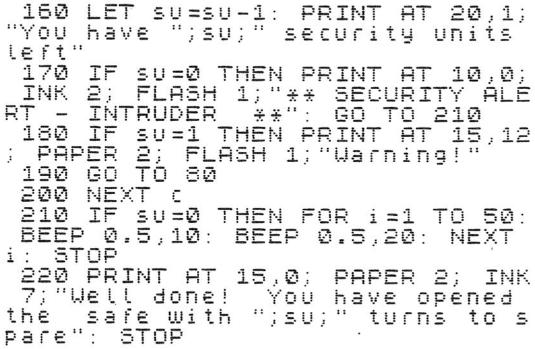
VARIABLES

l=line counter to initialize screen
array b dimension 50 = tone values for BEEP when dial moves
x = dial number

su=security units (i.e. attempts) left
cx,cy=x,y co-ordinates of center of dial circles
r=radius counter for drawing circles of dial
i,k=general purpose counters
c=dial section counter (1-3)
n=random combination code
d=player's estimate

## PROGRAM LISTING

```
   5 PAPER 5: FOR l=1 TO 21: PRI
NT TAB 31: NEXT l
   7 FOR l=3 TO 18: PRINT BRIGHT
 1; PAPER 7;AT l,8;"            
    ": NEXT l
   8 PLOT 64,23: DRAW 0,128: DRA
W 127,0: DRAW 0,-128: DRAW -127,
0
  10 DIM b(50): LET x=50: LET su
=10: LET cx=127: LET cy=87
  15 FOR r=25 TO 45: CIRCLE INK 
1,cx,cy,r: NEXT r
  20 FOR r=56 TO 64: CIRCLE INK 
0,cx,cy,r: NEXT r: INVERSE 1
  30 PRINT AT 0,0;"     CRACK TH
E COMBINATION      "
  35 PRINT AT 21,0;"            
                    ": INVERSE 0
  40 FOR i=1 TO 20: PRINT AT i,0
;"█";AT i,31;"█": NEXT i
  45 PRINT AT 3,15;50
  50 FOR c=1 TO 3
  60 LET n=INT (RND*46+5)
  70 FOR i=1 TO 50: LET b(i)=25-
(i<n)*(n-i)/2-(i>n)*(i-n)/2: NEX
T i
  75 LET b(n)=-10
  80 INPUT FLASH 1;"Enter next n
umber:";d
  90 LET d=INT d: IF d<1 OR d>50
 OR d=x THEN PRINT INK 2; FLASH 
1;AT 10,7;"UNAUTHORIZED ENTRY":
LET su=0: GO TO 210
 100 PRINT PAPER 4; FLASH 1;AT 3
,15;"--"
 110 FOR k=x TO d STEP 1-2*(d<x)
: BEEP 0.1,b(k)
 115 NEXT k
 120 PRINT AT 3,15;d: LET x=d
 125 IF d<10 THEN PRINT AT 3,15;
0;d
 130 IF d<>n THEN GO TO 160
 140 FOR i=1 TO c: BEEP 0.5,-5*i
: NEXT i
 150 PRINT AT 10,7;"Section ";c;
" unlocked": GO TO 200
```



```
 160 LET su=su-1: PRINT AT 20,1;
"You have ";su;" security units
left"
 170 IF su=0 THEN PRINT AT 10,0;
 INK 2; FLASH 1;"** SECURITY ALE
RT - INTRUDER **": GO TO 210
 180 IF su=1 THEN PRINT AT 15,12
; PAPER 2; FLASH 1;"Warning!"
 190 GO TO 80
 200 NEXT c
 210 IF su=0 THEN FOR i=1 TO 50:
 BEEP 0.5,10: BEEP 0.5,20: NEXT
i: STOP
 220 PRINT AT 15,0; PAPER 2; INK
 7;"Well done! You have opened
the safe with ";su;" turns to s
pare": STOP
```

NOTES

1. Step 2(b) of the program method, lines 70 and 75, sets up array b containing "BEEP tones." This illustrates an important principle in games programming, namely, do not waste time calculating when the player is waiting for a response—instead set up a table of values which can be looked up quickly. The values in array b are the numbers used in the BEEP instruction while the dial turns. There is a BEEP for each number that the dial is turned through, and these BEEP's must rise as the hidden code number is approached. There should be a low buzz when it is passed, and then BEEP's should fall as the dial is turned away from the number. Now it is quite feasible to calculate the size of BEEP's while the dial is being turned (line 110) and thus dispense with array b. However this would slow down the sound of the dial turning and therefore decrease the game's attractiveness.

2. It is helpful to include a warning message such as at step 2(c)(iii)C (line 180) when the player is getting close to losing.

## 3.2 MONEY MATCH

DESCRIPTION

This program is based upon a popular casino game. The computer chooses twenty numbers between 0 and 80 at random and the player has to guess what eight of these will be. Thus we have another of the "guess the number" family games.

The player starts with $20 and can gamble with part or all of it (in whole dollars). The payout is based upon the amount of the bet and how many numbers are right:

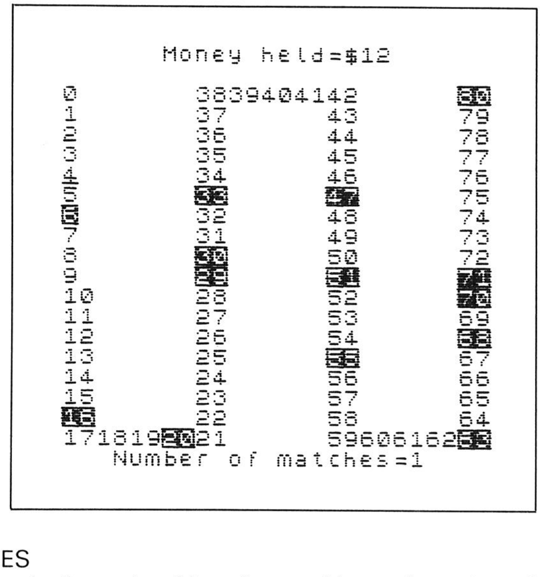
four numbers means payout = bet
five numbers means payout = 10 x bet
six numbers means payout = 100 x bet
seven numbers means payout = 1000 x bet
eight numbers means payout = 10000 x bet

The numbers 0 to 80 are shown on the screen in a pattern, and having specified the bet, the player enters eight "markers" or guesses as to what numbers the computer will choose. Next the computer chooses its twenty numbers in turn, each number 'bubbling' through the screen pattern. A match is highlighted and scored, and at the end the payout is shown. The game continues until the player runs out of money or chooses to stop.

METHOD

```
1.  Initialize screen and variables
    a  Set up screen positions of number 0 to 80 in arrays l
       (line) and c (column)
    b) Set money m to 20                                  (300)
2.  Display screen pattern including money held      (305-315)
3.  Set matches z to 0                                    (320)
4.  Enter bet b                                           (330)
5.  If b is invalid (a) Reenter                           (340)
                    (b) Go to (5)
6.  Decrease money m by bet b and display m               (350)
7.  Repeat for counter i from 1 to 8                      (360)
    a) Enter marker m(i)                                  (370)
    b) If m(i) is invalid (i) Reenter                     (375)
                          (ii) Go to (b)
    c) If m(i) is a duplicate (i) Reenter             (380-400)
                              (ii) Go to (c)
    d) Highlight marker m(i) on screen                    (410)
    e) Repeat to (7)                                      (420)
8.  Repeat for counter n from 1 to 20                     (430)
    a) User presses ENTER                                 (440)
    b) Calculate random number r(n)                       (450)
    c) If r(n) already chosen then go to (b)          (460-480)
    d) 'Bubble' r(n) through the screen pattern with
       sound                                          (490-495)
    e)  If r(n) = any of the markers m(1) to m(8) then       (500-510)
        (i)  add 1 to matches z
        (ii) sound confirmation
        (iii) display matches z                                  (520)
    f)  Repeat to (8)                                            (530)
 9. Calculate and display payout p                           (540-560)
10. Calculate resulting money held m and display m               (560)
11. If no money left (a) display message                         (560)
                     (b) stop
12. Enter choice to continue or stop                             (570)
    a)  If continue, go to (2)                                   (580)
    b)  If stop (i)  display message                             (570)
                (ii) stop
```

SCREEN LAYOUT



```
      Money held=$12

0       3839404142      80
1       37      43      79
2       36      44      78
3       35      45      77
4       34      46      76
5       33      47      75
6       32      48      74
7       31      49      73
8       30      50      72
9       29      51      71
10      28      52      70
11      27      53      69
12      26      54      68
13      25      55      67
14      24      56      66
15      23      57      65
16      22      58      64
1718192021      5960616263
   Number of matches=1
```

VARIABLES

array l, dimension 81 = line positions of numbers 0-80
array c, dimension 81 = column positions of numbers 0-80
eg. number 16 displayed at line l(17), column l(17)
i,j = general counters
m=money held, in $
z = no. of matches between user's markers and computer's numbers
b = user's bet

array m, dimension 8 = markers (or guesses) of user
n = counter showing sequence of computer's numbers
array r, dimension 20 = computer's random numbers
p=payout in $
z$ = number to specify continue ("1") or stop ("0")

PROGRAM LISTING

```
  15 DIM l(81): DIM c(81): DIM m
(8): DIM r(20)
  20 FOR i=1 TO 81: READ l(i),c(
i): NEXT i
  30 DATA 2,3,3,3,4,3,5,3,6,3,7,
3,8,3,9,3,10,3,11,3,12,3,13,3,14
,3,15,3,16,3,17,3,18,3,19,3,19,5
,19,7,19,9,19,11,18,11,17,11,16,
11,15,11,14,11,13,11,12,11,11,11
,10,11,9,11,8,11,7,11,6,11,5,11,
4,11,3,11,2,11,2,13,2,15,2,17,2,
19,3,19,4,19,5,19,6,19,7,19,8,19
,9,19,10,19,11,19,12,19,13,19,14
,19,15,19,16,19,17,19,18,19,19,1
9,19,21,19,23,19,25,19,27,18,27,
17,27,16,27,15,27,14,27,13,27,12
,27,11,27,10,27,9,27,8,27,7,27,6
,27,5,27,4,27,3,27,2,27
 300 LET m=20
 305 PAPER 7: BORDER 7: INK 0: C
LS : PRINT AT 0,9; INK 3;"Money
held=$";m
 310 FOR i=1 TO 81
 315 PRINT AT l(i),c(i); PAPER 7
; INK 0;i-1: NEXT i
 320 LET z=0
 330 INPUT "Enter your bet: $";b
 340 IF b>m OR b<>INT b OR b<1 T
HEN INPUT (b;" is invalid.  Reen
ter bet: $");b: GO TO 340
 350 LET m=m-b: PRINT AT 0,21;"
          ";AT 0,21; INK 3;m
 360 FOR i=1 TO 8
 370 INPUT ("Enter marker no.";i
;": ");m(i)
 375 IF m(i)<>INT m(i) OR m(i)<0
 OR m(i)>80 THEN INPUT "0 to 80
please: ";m(i): GO TO 375
 380 IF i=1 THEN GO TO 410
 390 FOR j=1 TO i-1: IF m(i)=m(j
) THEN INPUT ("You have ";m(i);"
 already.  Reenter:");m(i): GO T
O 390
 400 NEXT j
 410 PRINT AT l(m(i)+1),c(m(i)+1
); FLASH 1; PAPER 7; INK 2;m(i):
 PAUSE 100: PRINT AT l(m(i)+1),c
(m(i)+1); FLASH 0; PAPER 7; INK
2;m(i)
```

```
 420 NEXT i
 430 FOR n=1 TO 20
 440 INPUT ("Press ENTER for my
number ";n); LINE z$
 450 LET r(n)=INT (81*RND)
 460 IF n=1 THEN GO TO 490
 470 FOR j=1 TO n-1: IF r(n)=r(j
) THEN GO TO 450
 480 NEXT j
 490 FOR i=0 TO r(n): LET o$="__
": IF i<10 THEN LET o$="_"
 491 PRINT AT l(i+1),c(i+1); PAP
ER 8; INK 8; OVER 1;o$: IF i=10
THEN LET o$="_"
 494 IF i>0 THEN PRINT AT l(i),c
(i); PAPER 8; INK 8; OVER 1;o$
 495 BEEP 0.1,10: NEXT i: PRINT
AT l(r(n)+1),c(r(n)+1); INK 8; I
NVERSE 1;r(n)
 500 FOR j=1 TO 8: IF r(n)=m(j)
THEN LET z=z+1: BEEP 0.2,15: BEE
P 0.2,20: BEEP 0.2,15: GO TO 520
 510 NEXT j
 520 PRINT AT 20,6; INK 4;"Numbe
r of matches=";z
 530 NEXT n
 540 IF z<4 THEN LET p=0: GO TO
560
 550 LET p=b*10↑(z-4)
 560 PRINT AT 21,0; INK 1;"PAYOU
T=$";p: LET m=m+p: PRINT AT 0,21
; INK 8;m: IF m=0 THEN PRINT AT
21,15; INK 2; FLASH 1;"You are b
roke!": STOP
 570 INPUT "Key 1 to continue, 0
 to stop"; LINE z$: IF z$="0" TH
EN PRINT AT 21,0; INK 1;"You lea
ve with $";m: STOP
 580 GO TO 305
```

NOTES

1. After examining the numbers in DATA line 30 (and especially after keying them all in) the reader will see a pattern. It is in fact relatively straightforward to replace this massive block of DATA with several program loops to display the MONEY MATCH board. In this area there is no right or wrong way— it is up to the reader's preference. The author tends to err on the side of easier program entry, and in fact it is slightly shorter to choose the way the program appears above.

2. The method used to "bubble" a random number through the list merits attention. This appears at lines 490-495. The underline

character (_) is printed OVER a number to highlight it and then it is printed OVER again to remove the highlighting. This method is better than using a different color for highlighting as it is more difficult to return the number to its original color afterwards, and marked numbers tend to disappear. Note also in this section the need to differentiate between single digit and double digit numbers for underlining.

## 3.3 RECOGNIZE THE ENEMY

### DESCRIPTION

The essence of this number game is again quite straightforward. The player's ability to quickly assess the number of a random collection of spots is being tested. Many educationalists believe that people can be taught to do this quite easily. Young children particularly seem to have abilities in this area. Once again the bones of the idea are considerably padded out to make it more fun.

You are the captain of the TS2068 rocketship engaged upon its ninety nine year mission, boldly going where no one has gone before. Suddenly you find yourself being attacked by enemy ships. Subspace radar tracks them and displays them momentarily on the screen; you must estimate their number and fire the appropriate number of phasars. You also have a number of energy units which either rise or fall depending upon your success in estimating and firing. If your estimate is correct all the enemies are destroyed and your energy rises by fifty units; if you are within five of the correct number the ships not hit will flee and your units rise by ten. If however you are more than five out on your estimate you are in trouble! If you are too low the enemy ships not hit by a phasar will fire back; similarly if you are too high the excessive phasars will cancel out an equivalent number of other phasars, resulting in enemies left who will fire back. In both these cases, your energy units decrease by the number of ships firing at you. There are five waves of enemies and you may be forced to quit if your energy gets too low or if you make a foolish error in specifying the number of phasars required.

### METHOD

```
1. Initialize screen and variables
   a) Draw screen format                          (10-30)
   b) Set energy units u to 50 and display           (40)
2. Repeat for go counter g from 1 to 5               (50)
   a) Clear view screen and message line             (55)
   b) Wait for 1-5 seconds (random)                  (57)
```

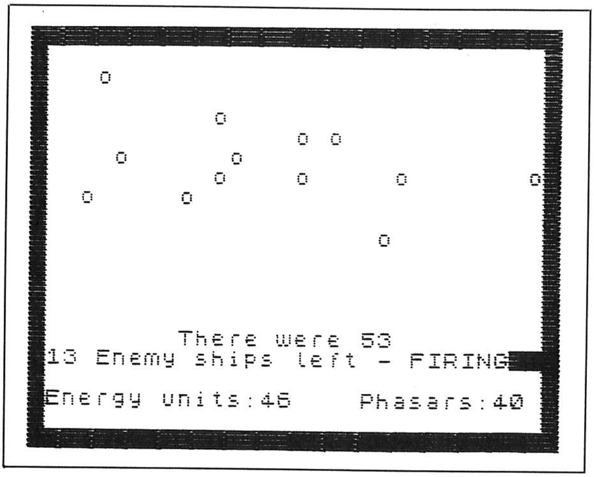
```
    c) Set no. of enemies r at random between
       20 and 99                                           (60)
    d) Display r enemies                              (SR 600)
    e) Display "ENEMIES SIGHTED"                           (70)
    f) Pause for 2 seconds
    g) Clear view screen and message line
    h) Enter no. of phasars e                               (80)
    i) If e is silly (i) Display message and claxon sound
                     (ii) Stop                      (90 and 300)
    j) Display no. of phasars to fire                       (95)
    k) Firing sound                                        (100)
    l) Display actual number r                             (105)
    m) If phasars e=enemies r                              (110)
       (i)  Display message
       (ii) Increase energy units u by 50              (SR 700)
       (iii) Go to (r)
    n) Calculate difference d between e and r              (120)
    o) Display d enemies                               (SR 600)
    p) If difference d < 6                                 (130)
         (i)  Display message
         (ii) Wait five seconds
         (iii) Clear view screen
         (iv) Increase energy units n by 10            (SR 700)
         (v)  Go to (r).
    q) Difference d > 6 so                                 (140)
         (i)  Display message
         (ii) Sound of firing
         (iii) Decrease energy units u by appropriate
               amount                          (150 and SR 700)
         (iv) If units u < 10                              (160)
               A) Display message and claxon sound         (300)
               B) Stop
    r) Press ENTER                                     (SR 800)
    s) Repeat to (2)                                       (170)
3.  End section
    a) Enter number z$                                     (200)
    b) If z$ = "1" go to (2)
    c) Stop                                                (210)
```

VARIABLES

i,j = general purpose counters
u = number of energy units
g = go number (1-5)

r=number of enemies
s = parameter to subroutine 600
e = number of phasars
d = difference between e and r and parameter to subroutine 700
z$ = dummy variable for ENTER
(l,c) = random line and column position used by subroutine 600

## SCREEN LAYOUT



```
             There were 53
13 Enemy ships left - FIRING
Energy units:46      Phasars:40
```

## SUBROUTINES

500 Displays one line of color
510 Displays border of color
520 Sets up (clears) viewer screen and message line
600 Displays s enemies on the viewer screen at random positions, each position different
700 Increases units u by d with display and sound
800 ENTER entry

## PROGRAM LISTING

```
  10 BORDER 7: PAPER 1: INK 5: G
O SUB 500: GO SUB 520
  20 PAPER 6: FOR i=1 TO 3: GO S
UB 510: NEXT i: GO SUB 500
```

```
  30 PRINT INK 0;AT 18,1;"Energy
 units:      Phasars:  "
  40 LET u=50: PRINT AT 18,14; I
NK 0; PAPER 6;u
  50 FOR g=1 TO 5
  55 GO SUB 520
  57 PAUSE INT (RND*5+1)*50
  60 LET r=INT (20+80*RND): LET
s=r: GO SUB 600
  70 PRINT INK 2; PAPER 7;AT 16,
1; FLASH 1;"       ENEMIES SIGHT
ED       ": PAUSE 100: GO SUB 52
0
  80 INPUT FLASH 1; INK 3;"Enter
 number of phasars to fire"'e
  90 IF e<>INT e OR e<10 OR e>99
 THEN PRINT AT 16,1; INK 2; PAPE
R 7; FLASH 1;"** ERROR!  Phasars
 jammed! **": GO TO 300
  95 PRINT AT 18,28;e
 100 FOR j=1 TO 5: FOR i=10 TO 2
0 STEP 2: BEEP 0.01,i: NEXT i: B
EEP 0.4,-20: NEXT j
 105 PRINT AT 15,9; INK 7; PAPER
 1;"There were ";r
 110 IF e=r THEN PRINT AT 16,1;"
       ABSOLUTELY ACCURATE    ":
 LET d=50: GO SUB 700: GO TO 170
 120 LET d=ABS (e-r): LET s=d: G
O SUB 600
 130 IF d<6 THEN PRINT AT 16,4;
PAPER 7; INK 2;d;" Enemy ships f
leeing ": PAUSE 250: GO SUB 520:
 LET d=10: GO SUB 700: GO TO 170
 135 IF d>r THEN LET d=r
 140 PRINT PAPER 7; INK 2;AT 16,
1;d;" Enemy ships left - "; FLAS
H 1;"FIRING": FOR i=1 TO 4: BEEP
 0.2,-20: BEEP 0.05,20: NEXT i
 150 LET d=-d: GO SUB 700
 160 IF u<10 THEN PRINT AT 18,14
; INK 1; PAPER 6; FLASH 1;u;AT 1
6,1; INK 2; PAPER 7;"ENERGY TOO
LOW - abandon ship!": GO TO 300
 170 GO SUB 800: NEXT g
 200 INPUT "The end.  Key 1 to c
ontinue"; LINE z$: IF z$="1" THE
N GO TO 50
 210 STOP
 300 FOR i=1 TO 20: BEEP 0.5,15:
 BEEP 0.5,5: NEXT i: CLS : BORDE
R 0: INK 2: PAPER 0: FOR i=0 TO
21: GO SUB 500: NEXT i: PAUSE 50
0: PAPER 7: INK 0: BORDER 7: STO
P
 500 PRINT "████████████████████
████████████": RETURN
 510 PRINT "█                   
           █": RETURN
 520 PAPER 1: INK 5: PRINT AT 0,
0: FOR i=1 TO 15: GO SUB 510: NE
XT i: PAPER 7: INK 2: GO SUB 500
: RETURN
 600 FOR i=1 TO s
 610 LET l=INT (1+RND*10): LET c
=INT (1+RND*30): IF SCREEN$ (l,c
)<>" " THEN GO TO 610
 620 PRINT PAPER 1; INK 6;AT l,c
;"o": NEXT i
 630 RETURN
 700 FOR i=u TO u+d STEP 1-2*(d<
0): PRINT AT 18,14; INK 1; PAPER
 6;i;"  ": BEEP 0.1,20: NEXT i: L
ET u=u+d: RETURN
 800 INPUT "press ENTER"; LINE z
$: RETURN
```

NOTES

1. The SCREEN$ function is hardly mentioned in the TS2068 manual, but it can be useful, as shown at line 610. Subroutine 600 is displaying a number of enemies on the screen, in random fashion. However one enemy ship must not be displayed on top of another, and ships must not appear over the edge of the televiewer. Thus we only print a ship where there is a blank on the screen. Thus to test whether a particular position on the screen (e.g. line 1, column c) is blank we simply need

IF SCREEN$ (l,c)="  " THEN ---

2. Subroutine 700 demonstrates a nice way of updating the display of energy units to their new value. Instead of just printing the new value on top of the old one, numbers are cycled through with a BEEP for each number, until the new value is reached. The slightly complicated looking STEP value is needed because the counting may be up (1) or down (-1)

# CHAPTER FOUR
# WORD GAMES

## 4.1 AWFUL ANAGRAMS

### DESCRIPTION

Many word games involve guessing what a word is. Awful Anagrams and Pace the Plank are both similar in method, but the former reveals the letters in the word as an anagram while the latter gives no such clues. It is a little tame to have a straightforward game shown on the TS2068 screen as it would be played on paper, so a little extra action is added.

You are the intrepid adventurer trapped by the wicked warlock in the cavern of riddles. In the cavern with you is a cauldron being heated over a fire. The only way out is through an enchanted door which opens by a magic word. This is the word shown on the screen in a jumbled form. You must guess the word letter by letter until you have it all and the door opens. However every time you make a mistake the cauldron's temperature rises, as shown by the thermometer. If it reaches the boiling point (200°) it will explode and take you, the enchanted door and the cavern with it.

The user may replace the words at DATA line 30 with words of their own choice, providing the number of words (n) is shown correctly at line 10.

### METHOD

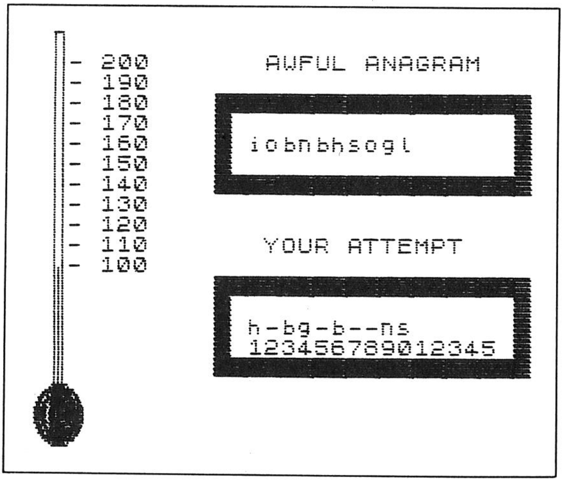
1. Initialize screen and variables
    a) Set n to no. of words available (10)
    b) Choose words w$ at random, based on limit of n (20-30)
    c) Set l to length of w$ (40)
    d) Set p$ and x$ to same size as w$
    e) Set p$ to all dashes
    f) Set letters-correct counter c to 0 (50)
    g) Set temperature t to 100

```
    h) Shuffle word w$ into x$                                    (50-60)
    i) Display screen format                                     (70-115)
2.  Clear message line                                              (120)
3.  Enter letter l$ and position p                                  (130)
4.  If p is invalid                                                 (140)
    a) Message
    b) Reenter p
    c) Go to (4)
5.  If letter l$ = letter p of w$                                   (150)
    a) add 1 to letters - correct c                                 (160)
    b) put l$ into position p of p$, part guessed so far
    c) display p$
    d) If letters - correct c = no. of letters l                    (170)
       (i) Display message of success
       (ii) go to (10)
    e) Go to (2)                                                    (180)
6.  Add 10 to temperature and display temperature rise              (190)
7.  If temperature t=200
    a) Display message of failure
    b) Sound of explosion and display
    c) Go to (10)
8.  If temperature t = 190                                          (200)
    a) Display warning message
    b) Wait for five seconds
9.  Go to (2)                                                       (210)
10. Enter number z$                                                 (220)
    a) If z$="1", rerun
    b) Stop
```

VARIABLES

n = no. of words in DATA statement
i = general purpose counter
w$ = word to be guessed
l = length of word w$
p$ = word correctly guessed so far, initially all dashes
x$ = anagram of w$
c = number of letters guessed correctly
t = temperature of cauldron
r = random number used to shuffle x$ and also radius counter
z$ = dummy character used in shuffling x$
l$ = letter entered by player
p = position of l— as entered by player
z$ = "1" to rerun program, "0" to stop

## SCREEN LAYOUT



## PROGRAM LISTING

```
   10 LET n=10
   20 RESTORE : FOR i=1 TO INT (1
+n*RND): READ w$: NEXT i
   30 DATA "hobgoblins","dragonsl
ayer","giantkiller","elfking","i
mmortality","infinity","oracle",
"elixir","magician","faeries"
   40 LET l=LEN w$: DIM p$(l): DI
M x$(l): FOR i=1 TO l: LET p$(i)
="-": NEXT i
   50 LET c=0: LET t=100: LET x$=
w$
   60 FOR i=1 TO l: LET r=INT (1+
l*RND): LET z$=x$(i): LET x$(i)=
x$(r): LET x$(r)=z$: NEXT i
   70 CLS : PAPER 7: BORDER 7: IN
K 2
   80 FOR r=1 TO 12: CIRCLE 20,24
,r: NEXT r
   90 PLOT 18,26: DRAW 0,149: DRA
W 4,0: DRAW 0,-149: PLOT 20,24:
DRAW 0,58
   95 INK 0: FOR i=1 TO 11: PRINT
 AT i,3;"- ";210-10*i: NEXT i
  100 INK 1: PRINT AT 1,15;"AWFUL
 ANAGRAM";AT 3,12;"█████████████
██████";AT 7,12;"███████████████
████": FOR i=4 TO 6: PRINT AT i,
12;"█";AT i,30;"█": NEXT i: PRIN
T AT 5,14;x$
```

```
 110 INK 3: PRINT AT 10,15;"YOUR
 ATTEMPT";AT 12,12;"████████████
███████";AT 16,12;"████████████
███████": FOR i=13 TO 15: PRINT A
T i,12;"█";AT i,30;"█": NEXT i:
PRINT AT 14,14;p$
 115 PRINT INK 5;AT 15,14;"12345
6789012345"
 120 INK 0: PRINT AT 18,12;"
                ";AT 19,12;"
                "
 130 INPUT "Enter letter: ";l$;"
, position:";p
 140 IF p<>INT p OR p<1 OR p>l O
R p>LEN w$ THEN INPUT (p;" is in
valid.  Reenter position:");p: G
O TO 140
 150 IF w$(p)<>l$ THEN GO TO 190
 160 LET c=c+1: LET p$(p)=l$: PR
INT AT 14,14; INK 3;p$
 170 IF c=l THEN PRINT AT 18,12;
"YOU ARE FREE!": GO TO 220
 180 GO TO 120
 190 LET t=t+10: DRAW INK 2;0,8:
 IF t=200 THEN PRINT AT 18,12;"Y
ou've failed - the";AT 19,12;"ca
uldron shakes...": FOR i=-25 TO
25: BEEP 0.1,i: NEXT i: PRINT AT
 0,0;: FOR i=1 TO 704: PRINT INV
ERSE 1;" ";: NEXT i: BORDER 2: G
O TO 220
 200 IF t=190 THEN PRINT AT 18,1
2; FLASH 1;"Cauldron is very hot
": PAUSE 250
 210 GO TO 120
 220 INPUT "key 1 to repeat, 0 t
o stop:"; LINE z$: IF z$="1" THE
N RUN
 225 BORDER 7: PAPER 7: INK 0: C
LS
 230 STOP
```

NOTES

1. The technique used to form an anagram of word w$ (lines 50 and 60) is very similar to the technique used in Chapter 8 to shuffle a pack of cards. It consists of going through the word letter by letter and swapping each letter with a letter in another (random) position. There are other ways of doing this shuffle, but this is the most simple and efficient.

2. Make sure the computer is in lower case when you run this program.

## 4.2 PACE THE PLANK

### DESCRIPTION

This is a Hangman variant in which the player has to guess a word, letter by letter. For every wrong letter chosen the player comes one step closer to death—in this case shown by a figure getting closer to the end of a plank suspended over water.

The word to be guessed is initially shown as a series of dashes. As a letter is guessed, it is shown in its correct position(s) in the word. If a wrong letter is guessed the letter is displayed under the wrong letters heading and the figure moves along the plank. After nine wrong attempts, the figure falls from the end of the plank. The user may replace the words at DATA line 100 with words of their own choice. (Be sure to use lower case when using this program.)

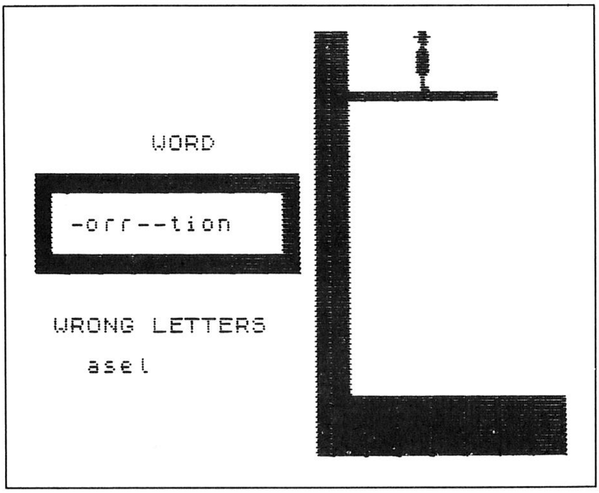
This is a good example of a simple concept which makes full use of TS2068 graphics and sound.

### METHOD

```
1.  Initialize screen and variables
    a) Set up graphic symbols A, B, C for figure          (10-50)
    b) Set no. of words n                                   (80)
    c) Choose word w$ at random                         (90-100)
    d) Set l to length of word w$                          (110)
    e) Set display pattern p$ to dashes, length l          (110)
    f) Set wrong letters string e$ to null                 (110)
    g) Set no. of correct letters c to zero                (120)
    h) Set position of man to 19                           (120)
    i) Draw screen format and headings                 (130-180)
2.  Clear message line                                     (190)
3.  Letter entry                                           (210)
    a) Enter user's chosen letter l$
    b) If letter l$ has already been chosen then       (220-260)
       (i) Reenter l$
       (ii) Go to (b)
4.  Test letter
    a) For counter k from l to length of w$                (270)
       (i) If letter l$ = letter k of w$ then
           A) Add 1 to no. of correct letters c            (280)
           B) Display letter l$ at position k of p$ on screen (280)
```

```
            C) If c=l then
                (I) Display CONGRATULATIONS                  (290)
                (II) Return to (1) or stop               (400-410)
        (ii) Repeat to 4(a)                                  (300)
    b) If letter l$ is not in word w$ then                   (305)
        (i) Add letter l$ to wrong letters string e$ and display (310)
        (ii) Add 1 to man's position m                       (310)
        (iii) Display man at column position m               (310)
        (iv) If m=28 then (A) Display man falling into sea   (320)
                          (B) Return to (1) at STOP          (400)
        (v) If m=27 then display warning message
    c) Go to (2)
```

SCREEN LAYOUT



VARIABLES

i,d = general purpose counters
a$ = holds A,B or C when setting up user-defined graphics
n = number of words available to choose from
w$ = word to be guessed
l = length of w$
p$ = word as displayed on the screen eg. —E—ES—
e$ =string of wrong letters i.e. letters specified by player which are not in word w$

c = number of correct letters chosen
m = column position at which man is displayed on line 0
l$ = letter chosen by player
ok = either 1, meaning l$ has not been chosen before, or
0, meaning l$ has been chosen before
inword = either 1, meaning l$ is in word w$, or
0, meaning l$ is not in word w$
k = position number of letter in w$ currently being compared with l$
z$ = "1" to repeat, "0" to stop.

PROGRAM LISTING

```
  10 FOR i=1 TO 3
  20 READ a$
  30 FOR d=0 TO 7: READ x: POKE
USR a$+d,x: NEXT d
  40 NEXT i
  50 DATA "A",60,60,255,60,62,28
,24,60,"B",126,126,126,126,126,1
26,126,126,"C",126,60,24,24,24,2
4,30,30
  80 LET n=10: RESTORE 100
  90 FOR i=1 TO INT (1+n*RND): R
EAD w$: NEXT i
 100 DATA "treachery","sorcery",
"espionage","mutiny","manslaught
er","murder","corruption","homic
ide","blasphemy","heresy"
 110 LET l=LEN w$: DIM p$(l): FO
R i=1 TO l: LET p$(i)="-": NEXT
i: LET e$=""
 120 LET c=0: LET m=19
 130 CLS : PAPER 7: BORDER 7: IN
K 1
 140 PRINT AT 5,7;"WORD";AT 7,0;
"████████████████";AT 11,0;"████
████████████";AT 9,2;p$: FOR i=8
 TO 10: PRINT AT i,0;"█";AT i,15
;"█": NEXT i
 150 PRINT INK 2;AT 14,1;"WRONG
LETTERS"
 160 FOR i=0 TO 17: PRINT INK 2;
 PAPER 5;AT i,17;"██";TAB 31;" "
: NEXT i: FOR i=18 TO 20: PRINT
INK 1;AT i,17;"███████████████":
 NEXT i
 170 PRINT INK 2; PAPER 5; INVER
SE 1;AT 3,19;"▄▄▄▄▄▄▄▄▄▄▄▄▄"
 180 PRINT INK 1; PAPER 5;AT 0,m
;"╋";AT 1,m;"█";AT 2,m;"╨"
 190 PRINT INK 0;AT 15,0;"
          "

 210 INPUT "Choose a letter: ";l
$
```

```
 215 PAPER 8
 220 LET ok=1: FOR i=1 TO LEN e$
: IF l$=e$(i) THEN LET ok=0
 230 NEXT i: FOR i=1 TO LEN p$:
IF l$=p$(i) THEN LET ok=0
 240 NEXT i
 250 IF NOT ok THEN INPUT (l$;"
has gone. Letter: ");l$: GO TO
220
 260 LET inword=0
 270 FOR k=1 TO l: IF w$(k)<>l$
THEN GO TO 300
 280 LET inword=1: LET c=c+1: LE
T p$(k)=l$: PRINT AT 9,2; INK 1;
p$
 290 IF c=l THEN PRINT AT 18,0;
INK 1; FLASH 1;"CONGRATULATIONS!
": GO TO 400
 300 NEXT k
 305 IF inword THEN GO TO 340
 310 LET e$=e$+l$: PRINT INK 2;A
T 16,3;e$: PRINT AT 0,m;" ♟";AT
1,m;" █";AT 2,m;" Ɪ": LET m=m+1
 320 IF m=28 THEN PRINT AT 18,0;
 INK 2;"One small step...": PRIN
T AT 20,0; PAPER 5; INK 1; FLASH
 1;w$: FOR i=0 TO 17: PRINT AT i
,28;" ";AT i+1,28;"♟";AT i+2,28;
"█";AT i+3,28;"Ɪ": BEEP 0.1,10-i
: NEXT i: GO TO 400
 330 IF m=27 THEN PRINT AT 18,0;
"Look out below!"
 340 GO TO 190
 400 INPUT "Key 1 to repeat, 0 t
o stop: "; LINE z$: IF z$="1" TH
EN RUN
 410 STOP
```

(Note: The characters **A, B** and **C** in lines 180, 310 and 320 must be entered in GRAPHICS mode (G cursor).)

NOTES

Two logical values are used in the program, namely ok and inword. Logical values, also known as flags or switches, can either be true or false, represented by 1 or 0 in the TS2068 (see Chapter 14 of the TS2068 manual) and can save time and increase program clarity if used in the right places.

## 4.3 SHRINK

DESCRIPTION

In the old days when computer games were played on teletypes connected to large mainframe computers, the Eliza program was developed. This was a very sophisticated program which simulated a psychiatrist interviewing a client. Shrink is a scaled-down version whereby the player keys in any thoughts which occur to him and the TS2068 gives a suitably analytical reply. It is a very useful demonstration of text analysis and each of the player's entries is processed in the following way. First keywords or phrases are sought e.g. "do you", "I want", "kill". Based upon the keyword found, a reply is chosen. However the reply may use part of the user's entry, in which case a certain amount of translation must be done, e.g. "I am" is changed to "you are", "my" to "your" and so on. Thus if the user enters "I want to kill you", the program may reply "Why do you want to kill me".

In playing the game it is best to use short sentences with no punctuation. **48K of memory is needed.**

METHOD

```
1.  Initialize variables and screen
    a) Set no. of keyboards kn and no. of substitution pairs
       sn                                                    (10)
    b) Set reply line nos. into r                         (20-26)
    c) Set colors                                            (30)

2.  Introduction
    a) Display welcome message                               (40)
    b) Enter user's name n$                                  (50)
    c) If first letter of n$ is not uppercase, make it
       uppercase                                          (52-55)
    d) Invite discussion                                    (60 )

3.  Repeat for questions q from 1 to 20:                     (70)
    a) Enter user's answer a$                                (80)
    b) If a$ = "stop" then stop                              (81)
    c) Display answer a$                                     (85)
    d) If last answer then (i) Display goodbye message   (86/370)
                          (ii) Stop                         (380)
    e) If answer is less than two chars. long
                            (i) set reply line number k to
                                'general' reply              (87)
                           (ii) go to (l)
    f) Convert upper case to lower case and omit apostrophes
       in answer a$                                      (90-100)
```

```
g) If answer a$ = previous answer b$
        (i) set reply line number k to 'repeat' reply      (110)
        (ii) go to (l)
h) Store answer a$ as b$                                    (120)
i) Put a full stop at end of answer a$                      (130)
j) Repeat for all the keywords (counter i from 1 to
   kn):—                                                    (140)
        (i)  If answer a$ contains keyword k$ (the ith
             keyword)                                  (170 -190)
                A) Set reply line number k to r(i)
                B) Set keyword position kp to position of
                   end of k$ in answer a$
                C) Go to (l)
        (ii) Repeat to (j)                                  (200)
k) Set reply line number k to 'general' reply               (210)
l) Get reply r$ based upon reply line number k              (300)
   (choose 1 of the 4 available, at random)
m) If last letter of reply r$ is * then                     (320)
        (i)  Remove last two letters of reply r$          (SR400)
        (ii) If keyword was at end of answer, go to (n)     (400)
        (iii) Set m$ to be the part of the answer a$
              following the keyword                         (410)
        (iv) Repeat for all substitution pairs              (420)
                A) Get word s$ and its replacement t$       (430)
                B) If s$ is in n$, then replace
                   it by t$                             (440-470)
                C) Repeat to (iv)                           (480)
        (v)  Put converted end m$ onto end of reply r$      (490)
n) If full stop is still present at end of reply, remove it
o) Display reply
p) Display name at end of reply, on a random basis
q) Repeat to (3)
```

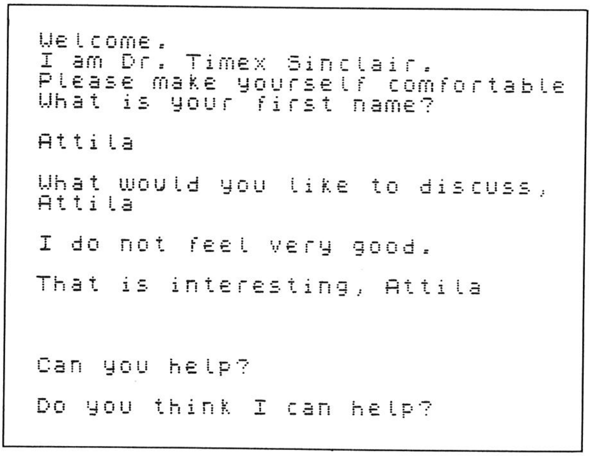
VARIABLES

kn = number of keywords
sn = number of substitution pairs
array r size kn = holds line nos. of DATA statements for each reply i.e. r(i) holds the line no. of the reply to a user's entry in which keyword number i is found
b$ = the previous answer
n$ = name of player
q = number of question/reply (1-20)
a$ = player's answer

k = line number of DATA statement holding appropriate reply
i,j = general purpose counters
ch = a character code number in the answer a$
l = length of answer a$
k$ = a keyword
kl = length of keyword k$
kp = position in a$ of end of keyword k$
r$ = reply
rl = length of reply
m$ = the part of the answer a$ after the keyword
s$,t$ = a substitution pair i.e. t$ is substituted for s$

## SCREEN LAYOUT

This simply consists of text displayed on the screen e.g.



```
Welcome.
I am Dr. Timex Sinclair.
Please make yourself comfortable
What is your first name?

Attila

What would you like to discuss,
Attila

I do not feel very good.

That is interesting, Attila


Can you help?

Do you think I can help?
```

## DATA

It is vital to understand how the data is arranged and used. Line 1000 contains the **key words and phrases** used to decide how to reply. The sequence that these are listed in is important. The reply to an answer containing "do you" is at line 3000, for "dont you" again it is 3000, for "can you" the reply is at line 3010 and so on. This list of reply line numbers is given in the DATA statements at lines 25 and 26. For example the fifth number is 3020. This means that for the fifth key word ("do I"), the reply is at line 3020.

Line 2000 contains **pairs of substitution words**. If a reply is going to incorporate part of the players answer e.g.

PLAYER: I want to kill you
TS2068: Why do you want to kill me?

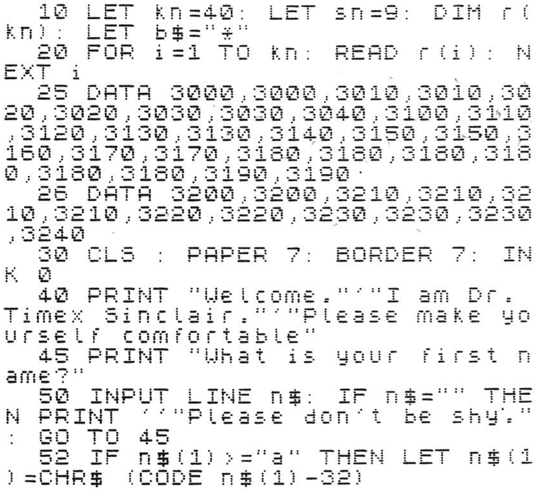
then "you" has to be translated to "me". Thus "I am" is changed to "you are" and so on. However to avoid things getting translated twice e.g.

you — me — you

a technical trick is used when keying in the program. **At the points indicated by an inverse space in line 2000 in the listing below, the user must put the TS2068 in E mode and then key 8. Nothing happens on the screen but a control character is placed in front of the substituted word so that it will not be retranslated.** For those interested, it happens to be the control character for 'normal brightness.

Lines 3000 onwards contain the **replies.** There are four replies on each line, and one is chosen at random, based upon what keyword is found in the user's entry. Some replies have an asterisk at the end. This indicates that part of the user's entry will be added at the end of the reply. Line 5000 has replies suited to repetitious answers and 5400 is used when no keyword is found. The game may be improved by adding more data according to the guidelines explained here.

## PROGRAM LISTING



```
  10 LET kn=40: LET sn=9: DIM r(
kn): LET b$="*"
  20 FOR i=1 TO kn: READ r(i): N
EXT i
  25 DATA 3000,3000,3010,3010,30
20,3020,3030,3030,3040,3100,3110
,3120,3130,3130,3140,3150,3150,3
160,3170,3170,3180,3180,3180,318
0,3180,3180,3190,3190
  26 DATA 3200,3200,3210,3210,32
10,3210,3220,3220,3230,3230,3230
,3240
  30 CLS : PAPER 7: BORDER 7: IN
K 0
  40 PRINT "Welcome."'"I am Dr.
Timex Sinclair."'"Please make yo
urself comfortable"
  45 PRINT "What is your first n
ame?"
  50 INPUT LINE n$: IF n$="" THE
N PRINT ''"Please don't be shy."
: GO TO 45
  52 IF n$(1)>="a" THEN LET n$(1
)=CHR$ (CODE n$(1)-32)
```

```
  55 PRINT 'n$''
  60 PRINT "What would you like 
to discuss, ";n$
  70 FOR q=1 TO 20
  80 INPUT LINE a$: IF a$="" THE
N PRINT ''"Try to tell me what's
 on your   mind, ";n$: GO TO 350
  81 IF a$="stop" THEN STOP
  85 PRINT 'a$''
  86 IF q=20 THEN GO TO 370
  87 IF LEN a$=1 THEN LET k=5400
: GO TO 300
  90 LET i=1
  92 LET ch=CODE a$(i): IF ch<=9
0 AND ch>=65 THEN LET a$(i)=CHR$
 (ch+32)
  95 IF a$(i)="'" THEN LET a$=a$
(1 TO i-1)+a$(i+1 TO )
 100 LET i=i+1: IF i<>LEN a$ THE
N GO TO 92
 110 IF a$=b$ THEN LET k=5000: G
O TO 300
 120 LET b$=a$
 125 RESTORE 1000
 130 LET a$=a$+".": LET l=LEN a$
 140 FOR i=1 TO kn: READ k$: LET
 kl=LEN k$
 170 FOR j=1 TO l-kl+1
 180 IF a$(j TO j+kl-1)=k$ THEN 
LET k=r(i): LET kp=j+kl: GO TO 3
00
 190 NEXT j
 200 NEXT i
 210 LET k=5400
 300 RESTORE k: FOR i=1 TO 1+RND
*4: READ r$: NEXT i
 320 LET rl=LEN r$: IF r$(rl)="*
" THEN GO SUB 400
 325 IF r$(LEN r$)="." THEN LET 
r$=r$(1 TO LEN r$-1)
 330 PRINT r$;
 340 IF RND>0.5 THEN PRINT ", ";
n$
 350 PRINT ''
 360 NEXT q
 370 PRINT INK 2;"Well, I'm afra
id that I have"'"another client 
waiting, ";n$;'"I've enjoyed tal
king to you."'"Goodbye."
 380 STOP
 400 LET r$=r$( TO rl-2): IF kp>
=l THEN RETURN
 410 RESTORE 2000: LET m$=a$(kp 
TO )
 420 FOR i=1 TO sn
 430 READ s$,t$
 440 LET sl=LEN s$: LET tl=LEN t
$: LET j=1
```

```
 445 IF j>LEN m$-sl+1 THEN GO TO
 480
 450 IF m$(j TO j+sl-1)=s$ THEN
LET m$=m$(1 TO j-1)+t$+m$(j+sl T
O ): LET j=j+tl-1
 460 LET j=j+1
 470 GO TO 445
 480 NEXT i
 490 LET r$=r$+m$
 500 RETURN
1000 DATA "do you","dont you","c
an you","cant you","do i","dont
i","can i","cant i","i feel","i
think","am i","are you","i canno
t","i cant","i can","im ","i am"
,"i want","because","so that","w
ho","why","when","where","how","
what","kill","hit ","die","dead"
,"ts2068","ts","compu","key","tv
","telev"," off","stop","shut ",
"sorry"
1500 DATA "f"
2000 DATA "i am","█you are█","yo
u are","█I am█"," you."," █me█",
"you ","█I "," me."," █you█"," i
 "," █you█ "," am "," █are█ ","
my"," █your█"," your"," █my█"
3000 DATA "Why do you ask whethe
r I *","Sometimes I do *","Would
 you like me to *","No I do not
*"
3010 DATA "Would you like me to
*","Can you *","Do you think I c
an *","Why do you ask this"
3020 DATA "Perhaps you do *","Wo
uld you like to *","What answer
would you like","Do you think yo
u will *"
3030 DATA "Do you want to *","Wo
uld you like to be able to *","I
s it normal to *","Have you aske
d anyone else this"
3040 DATA "Please explain these
feelings","Do you like feeling t
his way","Many people feel this
way","Do you often feel *"
3100 DATA "Are you sure","Why do
 you think *","You seem uncertai
n","What makes you think *"
3110 DATA "Do you want to be *",
"I think you may be *","Do you o
ften ask yourself this","Is this
 important to you"
3120 DATA "I would prefer not to
 say","Tell me whether you are *
","Do you like to think that I a
m *","Perhaps you think I am *"
```

love, cry, laugh,
[illegible]

H[illegible], [illegible], GOOD, Friend,

```
3130 DATA "Why cant you","Are yo
u sure you cant *","Would you li
ke to *","Is this normal for you
"
3140 DATA "Can you really","Can
your friends *","Do you like bei
ng able to *","Can you always *"
3150 DATA "Have you always been
*","Do you dream about being *",
"Do you want me to be *","Do you
 enjoy being *"
3160 DATA "Why do you want *","H
ow long have you wanted this","D
o you think it is normal to    w
ant *","How much do you want thi
s"
3170 DATA "Is this the only reas
on","Are there other reasons","I
s this really why","Do you belie
ve this is the real  reason"
3180 DATA "Let,s forget that. W
hy are you  asking questions","H
ow would you like me to answer
that","Does the question worry y
ou","Do you want to sit in my ch
air"
3190 DATA "Do you like violence"
,"Do you have violent fantasies"
,"Please calm yourself","I have
a pistol in my drawer"
3200 DATA "Does death interest y
ou","Do you like graveyards","Do
 you have morbid dreams","Are yo
u often morbid"
3210 DATA "Do you like computers
","Are you frightened of machine
s","What do you do when you are
not  here","What color are compu
ters in     your dreams"
3220 DATA "Are you keen on TV","
Do you find TV boring","Is your
TV big","Would you prefer to be
watching   Crossroads"
3230 DATA "I will ignore that","
Your time is not yet up","Do you
 like being aggressive","I am tr
ained to be tolerant"
3240 DATA "That's OK","don't apo
logize","Why are you sorry","Not
 at all"
5000 DATA "You just said that","
you are repeating yourself","Why
 do you repeat yourself","Please
 don't repeat yourself"
5400 DATA "Can you be more speci
fic","I see - go on","That is in
teresting","Do you feel happy ab
out this    discussion"
```

[illegible]

**Note: In line 2000, at the points marked with inverse spaces, enter E mode and press 8, as described in the DATA section above.**

NOTES

Finding and replacing words or phrases is fundamental to this program, so the techniques for carrying out these operations are now more fully described.

Consider the problem of finding the word DREAM in a sentence s$. DREAM has five letters so every five-letter sequence of the sentence must be examined. If we assume the sentence has 20 letters we therefore must examine the sequences:

s$(1 TO 5), s$(2 TO 6), s$(3 TO 7), ..., s$(16 TO 20) or more generally s$(i TO i+4) with i=1,2,3, ..., 16

For a sentence s$ of length **sl** this generalizes to looking at
s$ (i TO i+4) with i=1,2,3,..., sl—4

In a completely general case where we are searching for a word of length **wl** we examine
s$(i TO i+wl—1) with i=1,2,3, ..., sl—wl+1
To check this, consider the last value of i and see that it means that the last sequence examined is s$(s—wl+1,**sl**)

Thus a construction such as that at lines 170-190 may be found.

Substituting for words or phrases is a bit more complicated, mainly because the new word may not be the same length as the old word. The general principle is:

1. Find the word to be replaced, say between sentence positions p1 and p2 inclusive
2. Change the sentence so that it becomes

Characters 1 to (p1-1) + new word + characters (p2+1) to end.

It is not possible to use a FOR-NEXT loop here because of problems in defining the limits of the FOR statement. The length of the sentence changes in the middle of the loop.

Lines 440-470 above show how a word s$ is replaced by t$ in the sentence m$.

# CHAPTER FIVE

# BOARD GAMES

## 5.1 COPYCAT

### DESCRIPTION

This game is based on the popular electronic toy, found in a variety of forms, which produces random collections of colors and sounds which the player then has to repeat.

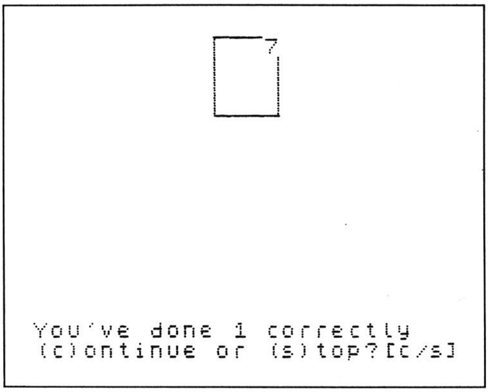
Use is made of the TS2068 color and sound facilities, but as it is difficult to enter a color or sound on the keyboard, the normal color numbers are used. Thus for each color, a block of color appears on the screen with the color number (e.g. 2 for red, 3 for magenta) on the top right. There is also a corresponding tone. The number of colors to be remembered and repeated rises progressively from 1 to 30.

### METHOD

```
1.  Set up screen and variables
    a) Set screen colors                                        (15)
    b) Generate a random sequence of color numbers (0-7) in
         array r size 30                                        (20)
2.  For sequence length n from 1 to 30                          (30)
    a) Press ENTER                                              (40)
    b) Produce color and sound sequence,
       length n                                 (50-80 and SR300)
    c) For counter c from 1 to sequence length n                (90)
       (i)   Enter color at position c (a(c))                  (100)
       (ii)  If a(c) is invalid (A) Reenter
                                (B) Go to (ii)
       (iii) Produce color a(c)                  (120 and SR 300)
       (iv)  If a(c) incorrect i.e. not=r(c) then              (130)
                 (i)  Sound error buzz
                 (ii) Go to 3(b)
       (v) Repeat to (c)                                       (140)
```

```
        d) Enter option z$                                  (150)
           (i) If z$ = "S", then stop
           (ii) Otherwise repeat to (2)                     (160)
    3.  End
        a) CONGRATULATIONS message                          (170)
        b) Reveal sequence of colors in array r         (187-210)
```

SCREEN LAYOUT



```
You've done 1 correctly
(c)ontinue or (s)top?[c/s]
```

VARIABLES

array r = holds random color numbers, size 30
array 1 = holds sequence of color number entered by player
i = general purpose counter
n = length of color sequence currently being shown
cn = color number supplied to subroutine 300
c = position of color in array r currently being considered
z$ = dummy value

SUBROUTINE

SR300 Displays a block of color, number cn, with cn at top right. Also gives an appropriate BEEP, depending on cn.

## PROGRAM LISTING

```
  10 DIM r(30): DIM a(30)
  15 PAPER 7: INK 0: BORDER 7: C
LS
  20 FOR i=1 TO 30: LET r(i)=INT
 (RND*8): NEXT i
  30 FOR n=1 TO 30
  40 INPUT "Press ENTER when you
 are ready"; LINE z$
  50 FOR c=1 TO n
  60 LET cn=r(c): GO SUB 300
  80 CLS : NEXT c
  90 FOR c=1 TO n
 100 INPUT ("What was color numb
er ";c;"?");a(c)
 110 IF a(c)<>INT a(c) OR a(c)>7
 OR a(c)<0 THEN INPUT "Between 0
 & 7.  Reenter:";a(c): GO TO 110
 120 LET cn=a(c): GO SUB 300
 130 IF a(c)<>r(c) THEN BEEP 2,-
20: GO TO 180
 140 NEXT c
 150 INPUT ("You've done ";n;" c
orrectly        (c)ontinue or (
s)top?[c/s]"); LINE z$: IF z$="s
" THEN STOP
 160 NEXT n
 170 PRINT AT 15,0;"Well done!
You've beaten me."
 185 PRINT AT 16,0;"The sequence
 was:"
 187 PAUSE 150
 190 IF n>30 THEN LET n=30
 200 FOR i=1 TO n: PRINT BRIGHT
1; PAPER r(i); INK 9;r(i);: BEEP
 0.5,r(i)*5-20: NEXT i
 210 STOP
 300 FOR i=6 TO 9: PRINT AT i,11
; BRIGHT 0; PAPER cn;"    ": NEX
T i
 305 PLOT 88,127: DRAW 0,-31: DR
AW 31,0: DRAW 0,31: DRAW -31,0
 310 PRINT AT 6,14; INK 9;cn: BE
EP 1,cn*5-20
 320 RETURN
```

## NOTES

Readers may notice that the color sequence entered by the player need not be stored in an array (**a** above) with the program as it stands. If however the program is required to be extended, a stored sequence probably will be required. For example one may wish to print out the player's entries at the end of the program to compare them with the actual code. Thus we have a case of a certain amount of redundancy built in to allow for future modifications.

## 5.2 WHAT'S THE CODE?

### DESCRIPTION

This is a sophisticated version of the famous board game with colored pegs in which a color code has to be found. The computer chooses the code, with between three and eight color positions, as specified by the player. Each of the player's attempts is shown and marked, in the normal way, with black or white pegs. A black peg corresponds to a color chosen correctly in the right position. A white peg indicates a color present but in the wrong position. To make colors easier to identify the color number is also shown in the colored square. The player has thirteen attempts to find the hidden code, after which it is revealed. The player can also specify at the beginning of the program whether or not certain colors will be repeated in the code.

### METHOD

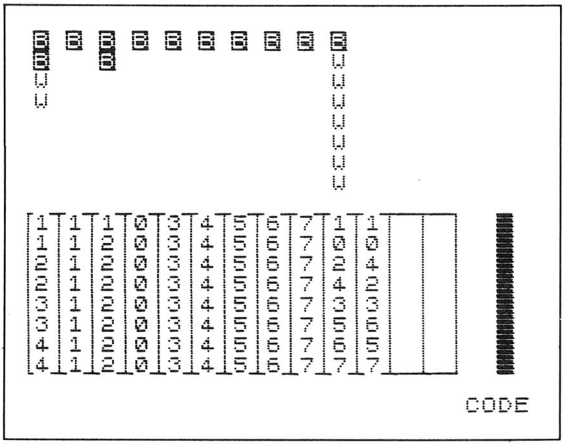
```
1.  Initialize screen and variables
    a) Set screen colors                                    (10)
    b) Enter no. of code positions p                        (20)
    c) If p invalid (i) Message and reenter                 (30)
                    (ii) Go to (c)
    d) Enter whether or not repeated colors required     (40-50)
    e) Draw screen format                                (60-80)
2.  Generate random code. For counter i from 1 to
    position p                                             (100)
    a) Set code position c(i) to random number (0-7)       (110)
    b) If repeats allowed or i = 1 go to (d)               (110)
    c) Otherwise if c(i) = any previous element of code
       array c then go to (a)                          (120-130)
    d) Repeat to (2)                                       (140)
3.  For go number g from 1 to 13                           (200)
    a) For counter i from 1 to position p                  (210)
       (i)   Enter players choice of color no. i into a(i) (220)
       (ii)  If a(i) invalid then (A) Message and reenter  (225)
                                  (B) Go to (ii)
       (iii) Display color and number a(i)                 (230)
       (iv)  Repeat to (a)                                 (240)
    b) Enter confirmation z$                               (250)
       (i) If z$  "  " then A) rub-out chosen code         (260)
                            B) Go to (a)
```

```
      c) Marking
         (i)  Set black (b) and white (w) counters to zero         (300)
         (ii) Copy code (array c) into array d                     (300)
         (iii) (BLACK MARKS) For counter i from 1 to
              positions p                                          (310)
              A) If user's choice a(i) = code d(i)                 (320)
                 (I)  add 1 to b
                 (II) set a(i) and d(i) to 9
              B) Repeat to (iii)                                   (330)
         (iv) (WHITE MARKS) For counter i from 1 to
              positions p                                          (340 )
              A) If user's choice a(i) = 9 then go to (c)          (350)
              B) For counter j from 1 to positions p               (360)
                 (I)  If user's choice a(i) = code d(j)            (370)
                        then add 1 to w and set a(i) and d(j) to 9
                 (II) repeat to (B)                                (380)
              C) Repeat to (iv)                                    (390)
         (v)  If black points b = no. of positions p then          (395)
              A) CONGRATULATIONS message
              B) Go to (5)
         (vi) Display b black marks and w white marks
              starting at line l = 0                          (400-430)
      d) Repeat to (3)                                             (450)
   4. Display OUT OF GUESSES message                               (460)
   5. Reveal code                                                  (500)
   6. Return to (1) or stop                                    (510-520)
```

VARIABLES

p = no. of code positions
array a = user's guess at code
array c = random code
array d = copy of c
z, z$ = dummy input variable
rep = 0 for repeats not allowed, or = 1 repeats allowed
l = line counter
c = column counter
i,j = general purpose counters
g = go number
w = white marks
b = black marks

## SCREEN LAYOUT



## PROGRAM LISTING

```
  10 PAPER 7: BORDER 7: INK 0: C
LS
  20 INPUT "How many code positi
ons (3-8):";p
  30 IF p<>INT p OR p<3 OR p>8 T
HEN INPUT (p;" is not between 3
& 8. Number:");p: GO TO 30
  35 DIM a(p): DIM c(p): DIM d(p
)
  40 INPUT "Do you want repeated
 colors?     (y/n):";z$
  50 LET rep=0: IF z$="y" THEN L
ET rep=1
  60 LET j=p*8-1: FOR i=28 TO 22
0 STEP 16: PLOT i,103: DRAW 0,-j
: NEXT i: DRAW -207,0: DRAW 0,j:
 DRAW 207,0
  70 FOR l=9 TO 9+p-1: PRINT AT
l,30;"█": NEXT l
  80 PRINT AT 10+p,28;"CODE"
 100 FOR i=1 TO p
 110 LET c(i)=INT (8*RND): IF re
p OR i=1 THEN GO TO 135
 120 FOR j=1 TO i-1: IF c(i)=c(j
) THEN GO TO 110
 130 NEXT j
 140 NEXT i
 200 FOR g=1 TO 13
 210 FOR i=1 TO p
 220 INPUT ("Enter color no. ";i
;": ");a(i)
```

```
 225 IF a(i)<>INT a(i) OR a(i)<0
 OR a(i)>7 THEN INPUT "Must be 0
 - 7.  Reenter: ";a(i): GO TO 22
5
 230 PRINT AT 8+i,g*2; BRIGHT 1;
 PAPER a(i); INK 9;a(i)
 240 NEXT i
 250 INPUT "Press * to cancel, a
ny other key to confirm:";z$
 260 IF z$="*" THEN FOR l=9 TO 8
+p: PRINT AT l,g*2;" ": NEXT l:
GO TO 210
 300 LET w=0: LET b=0: FOR i=1 T
O p: LET d(i)=c(i): NEXT i
 310 FOR i=1 TO p
 320 IF a(i)=d(i) THEN LET b=b+1
: LET a(i)=9: LET d(i)=9
 330 NEXT i
 340 FOR i=1 TO p
 350 IF a(i)=9 THEN GO TO 390
 360 FOR j=1 TO p
 370 IF a(i)=d(j) THEN LET w=w+1
: LET a(i)=9: LET d(j)=9: GO TO
390
 380 NEXT j
 390 NEXT i
 395 IF b=p THEN PRINT AT 20,0;
INK 3; FLASH 1;"You've got it in
 ";g;" goes!": GO TO 500
 400 LET l=0
 405 IF b=0 THEN GO TO 420
 410 PRINT AT l,g*2; INVERSE 1;"
B": LET l=l+1: LET b=b-1: GO TO
405
 420 IF w=0 THEN GO TO 440
 430 PRINT AT l,g*2; BRIGHT 1;"W
": LET l=l+1: LET w=w-1: GO TO 4
20
 450 NEXT g
 460 PRINT AT 19,0;"Sorry.  You
have not guessed it"'"It is show
n above"
 500 FOR i=1 TO p: PRINT AT 8+i,
30; BRIGHT 1; PAPER c(i); INK 9;
c(i): NEXT i
 510 INPUT "Enter 1 to repeat, 0
 to stop",z: IF z=1 THEN RUN
 520 STOP
```

## NOTES

In this listing the marking strategy is important. This is shown at steps c(iii) and (iv) or program lines 310 to 390. The **black** marks must be worked out first and any matching pegs must be ignored when moving on to white marking. Similarly a peg must not be allowed to count for more than one **white** mark. To ensure this, a color is set to 9 when it has contributed to either a black or white point in the marking.

## 5.3 SOLITAIRE

DESCRIPTION

This is the familiar board game using 33 peg positions with the middle peg removed. The objective is to end up with one peg left (preferably in the middle position) by means of jumping one peg over another, the jumped peg being removed. This TS2068 version allows legal horizontal, vertical **and** diagonal jumps. The player specifies positions to move from and to by entering a pair of row and column positions. To allow for mistakes or for stalemate positions two extra facilities are available: if the player enters a zero for the row or column position of the "from" position, the game ends; if zero is entered for the "to" position the current "from" entry is cancelled and the complete move may be reentered.

METHOD

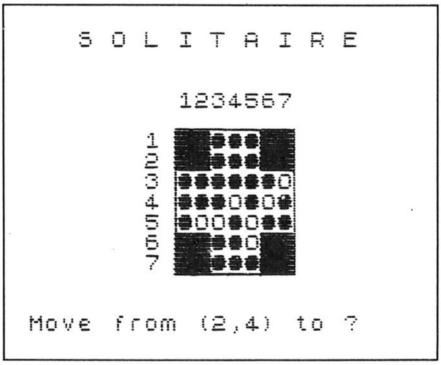
```
1.  Initialize screen and variables
    a) Screen colors and clear screen                          (10)
    b) Set up graphic for peg (p)                           (20-30)
    c) Set up array b to hold board                         (40-50)
    d) Print titles                                         (60-70)
    e) Display board with middle peg removed               (80-120)
    f) Set no. of pegs left (np) to 32                        (130)
2.  Player's go
    a) Clear message lines                                    (150)
    b) Enter move:                                            (160)
       (i)   Enter position from which to move (fr,fc)        (170)
       (ii)  If value = 0 then                                (175)
                A) STOPPED message, including pegs left (np)
                B) Go to 3
       (iii) If fr or fc invalid then                         (180)
                A) Message and reenter
                B) Go to 2(b)(ii)
       (iv)  Display 'from' position (fr,fc)                  (190)
       (v)   Enter position to which to move (tr,tc)          (200)
       (vi)  If value = 0 then go to 2                        (210)
```

```
        (vii) If fr or fc invalid then                              (215)
                A) Message and reenter
                B) Go to 2(b)(vii)
        (viii) Calculate column increment (ic) and row
                    increment (ir)                                  (220)
        (ix) Calculate signs of increments (sc and sr)              (220)
        (x)  If move invalid (ic, ir not equal to 2 or 0) then  (230-250)
               (A) Message (NO SUCH JUMP) and reenter
               (B) Go to 2(b)(vi)
        (xi) Calculate position of jumped peg (jr, jc) using
               sr and sc                                            (260)
        (xii) If no peg at this position then                       (270)
                (A) Message (NO SUCH JUMP) and reenter
                (B) Go to 2(b)(vi)
     c) Display 'to' position (tr,tc)                               (280)
     d) Effect move in array b                                      (290)
     e) Display move on board with sound                            (300)
     f) Decrement pegs left (np) by 1                               (310)
     g) If only one peg left then                                   (310)
        (i) Congratulatory message and sound
        (ii) Go to 3
     h) Go back to 2                                                (320)
3.   End
     a) Rerun or stop                                           (330-340)
```

VARIABLES

i,j = general purpose counters  
x = pattern for peg graphic; also parameter for FN a  
array b = holds board (7 x 7)  
Value 1 denotes peg, value −1 denotes hole, and value 0 denotes invalid position  
np = number of pegs left on board  
y = parameter for FN c  
fr, fc = position to move from  
tr, tc = position to move to  
jr, jc = jumped peg position  
ir = difference between tr and fr (row increment)  
ic = difference between tc and fc (column increment)  
sr = signum (SGN) of ir (ie. −1 if negative, 1 if positive, 0 if0)  
sc = signum (SGN) of ic

## SCREEN LAYOUT



```
S O L I T A I R E

1234567

Move from (2,4) to ?
```

## FUNCTIONS

FN a = true if parameter x is non-integer or outside the range 1-7, i.e. an invalid row or column entry

FN c = gives character code 159 (peg graphic) if parameter y is 1, and code 79 (hole) if y is −1. Used to display board using board array b.

## PROGRAM LISTING

```
  10 PAPER 7: BORDER 7: INK 0: F
LASH 0: CLS
  15 DEF FN a(x)=((x<>INT x) OR
(x<1) OR (x>7)): DEF FN c(y)=((y
=1)*159+(y=-1)*79)
  20 FOR i=0 TO 7: READ x: POKE
USR "P"+i,x: NEXT i
  30 DATA 0,60,126,126,126,126,6
0,0
  40 DIM b(7,7): FOR i=1 TO 7: F
OR j=1 TO 7: READ b(i,j): NEXT j
: NEXT i
  50 DATA 0,0,1,1,1,0,0,0,0,1,1,
1,0,0,1,1,1,1,1,1,1,1,1,-1,1,1
,1,1,1,1,1,1,1,1,0,0,1,1,1,0,0,0
,0,1,1,1,0,0
  60 PRINT AT 0,7; INK 1;"S O L
I T A I R E"; INK 0;AT 3,13;"123
4567"
  70 FOR i=1 TO 7: PRINT AT i+4,
11;i: NEXT i
  80 FOR i=1 TO 7: PRINT AT i+4,
13;: FOR j=1 TO 7
  90 IF b(i,j)=0 THEN PRINT "▒";
: GO TO 110
 100 PRINT PAPER 4; INK 7;CHR$ (
FN c(b(i,j)));
 110 NEXT j: NEXT i
 120 PLOT 103,136: DRAW 57,0: DR
AW 0,-57: DRAW -57,0: DRAW 0,57
 130 LET np=32
 150 PRINT AT 14,0;"
                    "
 155 PAUSE 100
 160 PRINT AT 14,4; INK 1;"Move
from ?";
 170 INPUT "row:";fr;" column:";
fc
 175 IF fr=0 OR fc=0 THEN PRINT
AT 18,1;"You stopped with ";np;"
 pegs left": BEEP 2,-20: GO TO 3
30
 180 IF FN a(fr) OR FN a(fc) OR
b(fr,fc)<>1 THEN INPUT "No - ree
nter: row=";fr;" col=";fc: GO TO
 175
 190 PRINT CHR$ 8;"(";fr;",";fc;
") to ?";
 200 INPUT "row:";tr;" column:";
tc
 210 IF tr=0 OR tc=0 THEN GO TO
150
 215 IF FN a(tr) OR FN a(tc) OR
b(tr,tc)<>-1 THEN INPUT "No - re
enter: row=";tr;" col=";tc: GO T
O 210
 220 LET ic=tc-fc: LET ir=tr-fr:
 LET sc=SGN (ic): LET sr=SGN (ir
)
 230 IF ABS (ic)=2 THEN IF ABS (
ir)=2 OR ABS (ir)=0 THEN GO TO 2
60
 240 IF ABS (ir)=2 THEN IF ABS (
ic)=0 THEN GO TO 260
 250 INPUT "No such jump - reent
er: row=";tr;"    col=";tc: GO T
O 210
 260 LET jc=fc+sc: LET jr=fr+sr
 270 IF b(jr,jc)<>1 THEN INPUT "
No peg to jump over - reenter:
row=";tr;" col=";tc: GO TO 210
 280 PRINT CHR$ 8;"(";tr;",";tc;
")."
 290 LET b(jr,jc)=-1: LET b(fr,f
c)=-1: LET b(tr,tc)=1
 300 PRINT PAPER 4; INK 7;AT fr+
4,fc+12;CHR$ (FN c(b(fr,fc)));AT
 tr+4,tc+12;CHR$ (FN c(b(tr,tc))
);AT jr+4,jc+12;CHR$ (FN c(b(jr,
jc))): BEEP .1,10: BEEP .1,15: B
EEP .1,20
```

```
 310 LET np=np-1: IF np=1 THEN P
RINT AT 18,1; INK 1; FLASH 1;"Yo
u have completed the puzzle!": B
EEP .5,15: BEEP .5,25: GO TO 330
 320 PAUSE 100: GO TO 150
 330 PAUSE 150: INPUT "Press ENT
ER to play again, or 0 to stop";
 LINE z$: IF z$="" THEN RUN
 340 STOP
```

NOTES

1. Much of the program is concerned with checking that the user's entry of a move is valid. Apart from checking whether there is a piece to be moved and a hole to be moved into, this validation involves calculating whether it is a correct two-place move in a vertical, horizontal or diagonal direction, and also whether there is a relevant peg to be jumped over in this direction. A valid move can be considered by looking at the differences between the 'to' and 'from' positions entered, in terms of the row difference (ir) and the column difference (ic). If the unsigned values of these are either 0 or 2, with at least one being non-zero, then it looks valid. However, the 'jumped' position must finally be checked to find whether it contains a peg. This is done by looking at the signs of the differences ir and ie, as shown in lines 220, 260 and 270.

2. The peg graphic character is displayed, not by an explicit PRINT "P", but by referring to the character code of graphic P, i.e. 159. This method facilitates the use of function (FN c above) which can print either a peg or a hole as appropriate.

# CHAPTER SIX
# SIMULATION GAMES

## 6.1 SINK THE SHIP

### DESCRIPTION

This short game is designed to show the principles of an interactive game in which the player has to react quickly and accurately. It resembles fairground games in which you try to hit a moving object by firing an air rifle. A ship moves left to right across the screen against a backdrop of the numbers 1 to 9. The player has to press the appropriate number when the ship is exactly under it to score a hit but it is not possible to win by just keeping a finger on a number key and waiting for the ship to pass by!

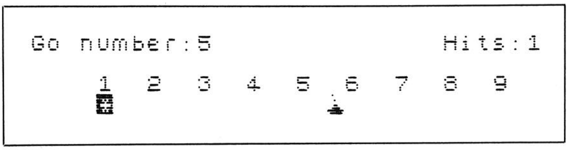
The speed of the ship's motion is governed by line 5. If a smaller number than 15 is entered, the ship moves more quickly, while a larger number makes the ship go more slowly.

### METHOD

```
1.  Initialize screen and variables
    a) Set wait period to 15                                  (5)
    b) Set up user-defined graphic for ship               (10-40)
    c) Set hits h to 0                                       (50)
    d) Set up screen colors                          (50 and 70)
    e) Display screen format and titles              (60 and 80)
2.  For go number g from 1 to 10                            (100)
    a) Set position p of ship to 1                          (110)
    b) Clear ship and message lines                         (110)
    c) Rub-out ship's previous position and display it at p (120)
    d) If keys 1-9 pressed then
       (i)   Set sp to position of shot (i.e. position of number
             on screen)
       (ii)  Display shot at this position
       (iii) If shot position sp = ship position p then     (150)
             A) Message and sound confirmation              (160)
```

```
            B) Add 1 to hits h and display h                  (160)
            C) Go to (i)                                      (160)
        e) If key still pressed go to (e)                     (180)
        f) Count up to wait number                            (185)
        g) Add 1 to ship position p                           (190)
        h) If ship position p≤31 then go to 2(c)              (190)
        i) Press ENTER                                        (200)
        j) Repeat to (2)                                      (210)
3.  Display number of hits and stop                       (220-230)
```

SCREEN LAYOUT



```
Go number:5                 Hits:1

       1  2  3  4  5  6  7  8  9
```

VARIABLES

wait = delay governing speed of ship's motion
c = counter used in setting up ship graphic symbol
x = bit pattern used in setting up ship graphic symbol
h = number of hits scored
i = general purpose counter
g = go number
p = position of ship - a column number on line 3
n$ = holds value of key pressed
sp = shot position - a column number on line 3
z$ = dummy variable

PROGRAM LISTING

```
   5 LET wait=15
  10 FOR c=0 TO 7
  20 READ x: POKE USR "S"+c,x
  30 NEXT c
  40 DATA 32,0,16,0,24,60,255,12
6
  50 LET h=0: CLS : BORDER 6: PA
PER 6: INK 0
  60 PRINT INK 2;"Go number:";TA
B 25;"Hits:  ": FOR i=1 TO 21: P
RINT PAPER 6-(i=2)*4-(i=3)-(i=4)
*5;TAB 31;" ": NEXT i
  70 PAPER 8: INK 9
  80 FOR i=1 TO 9: PRINT AT 2,3*
i+1;i: NEXT i
```

```
 100 FOR g=1 TO 10
 110 LET p=1: PRINT AT 0,10;g;AT
 3,0;"
          ";AT 6,0;"
                    "
 120 PRINT AT 3,p-1;" S"
 130 LET n$=INKEY$: IF n$>"9" OR
 n$<"1" THEN GO TO 180
 135 LET sp=3*VAL n$+1
 140 PRINT AT 3,sp; INVERSE 1;"*
": BEEP 0.2,-20
 150 IF sp<>p THEN GO TO 180
 160 PRINT AT 6,10; FLASH 1;"DIR
ECT HIT!": BEEP 0.2,10: BEEP 0.2
,20: BEEP 0.2,15: LET h=h+1: PRI
NT AT 0,30;h: GO TO 200
 180 IF INKEY$<>"" THEN GO TO 18
0
 185 FOR i=1 TO wait: NEXT i
 190 LET p=p+1: IF p<=31 THEN GO
 TO 120
 200 INPUT "Press ENTER to conti
nue:"; LINE z$
 210 NEXT g
 220 PRINT AT 6,0;"That's it.  Y
ou hit ";h;" out of 9"
 230 STOP
```

**(Note: The S at line 120 must be entered in graphics mode (G cursor).)**

NOTES

At first line 180 may appear unnecessary, but it is needed to prevent cheating. Without this line the player could keep a key down and keep shooting, so that the ship was bound to be hit when it passed the number.

## 6.2 HEADQUARTERS

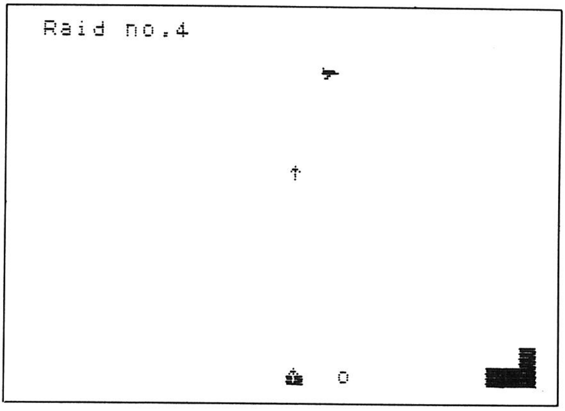
DESCRIPTION

Headquarters is a sophisticated development of the concept in Sink the Ship. The player is in charge of a missile launcher which he moves left (key 5) or right (key 8) and can fire (key 7). The launcher has to protect the player's headquarters bunker which is being attacked by enemy bombers. These bombers appear from the left of the screen and drop their darts of destruction with (almost) unerring accuracy. Ten bombers can appear, and if the headquarters suffers six hits it is destroyed.

METHOD

```
1. Initialize screen and variables
   a) Set up positions for plane, ground and HQ                (10)
   b) Set bomb as not dropped and missile as not fired          (10)
   c) Set HQ building to undamaged (h$)                         (10)
   d) Set planes hit (p) to 0 and HQ hits(h) to 0               (10)
   e) Set up user graphic for plane                          (20-30)
   f) Draw screen format                                     (40-60)
2. For raid number g from 1 to 10                              (100)
   a) Display g                                                (105)
   b) Clear ground lines and plane line                        (110)
   c) Set initial plane (pc) and launcher (lc) column
      positions                                                (120)
   d) Add 1 to plane position (pc)                             (130)
   e) If plane at RH side of screen go to (m)
   f) (DROP A BOMB?) If bomb not dropped then                  (140)
      (i) If plane at correct position or 3% chance then
          A) Drop bomb
          B) Set bomb line (bl) and column (bc) positions
   g) Display plane                                            (145)
   h) Display launcher                                         (145)
   i) If bomb dropped then                                     (150)
      (i) Rub-out old bomb position and set up new
          bomb position                                        (160)
      (ii) If bomb hits HQ                                     (170)
          A) Display HQ (h$) flashing with sound of explosion
          B) Add 1 to HQ hits (h)
          C) Remove part of HQ and redisplay                   (180)
          D) Set bomb as not dropped
          E) If hits h=6 then                                  (190)
             (I) Message of defeat
             (II) Go to 4                                      (210)
      (iii) If bomb hits ground then display explosion with
          sound                                            (220-230)
   j) If missile fired                                         (250)
      (i) Rub-out old missile and set up new missile
          position                                             (260)
      (ii) If missile below plane then go to (k)               (260)
      (iii) If missile missed plane then
          A) Display and sound explosion                       (280)
          B) Set missile as not fired                          (280)
          C) Go to (k)                                         (280)
      (iv) (PLANE HIT) Display and sound hit on plane,
           and message                                          (290 )
      (v)  Go to (m)
    k) Test key pressed                                         (300)
      (i)   If no key go to 2(d)                                (300)
      (ii)  If key = 5 then move launcher left (if possible)    (310)
      (iii) If key = 8 then move launcher right (if possible)   (320)
      (iv)  If key = 7 then                                     (330)
              A) If missile not fired then
                   (I)  fire missile
                   (II) set missile position
    l) Go back to (d)                                           (340)
    m) Press ENTER                                              (350)
    n) Rub-out old plane and launcher positions                 (355)
    o) If missile fired then rub-out missile and set to
       not fired                                                (356)
    p) If bomb dropped then rub-out bomb (if unexploded)
       and set to not dropped                                   (357)
    q) Repeat to (2)                                            (360)
3.  (HQ OK) Message of congratulations                          (380)
4.  Rerun to 1 or stop.                                         (400)
```

SCREEN FORMAT



Raid no.4

0

VARIABLES

pl,pc = plane line and column position
hqc = HQ column positions
gl = ground line position

bl,bc = bomb line and column position
ml,mc = missile line and column position
dropped = bomb dropped (=1) or not dropped (=0)
fired = missile fired (=1) or not fired (=0)
h = number of hits on HQ
p = number of planes hit
i = general purpose counter
x = bit pattern used in setting up plane graphic
h$ = string used to display HQ at various stages of destruction
n$ = value of key pressed
l = line number used by subroutine 650

## SUBROUTINES

SR600 Displays HQ
SR650 Clear line l
SR700 Sound explosion

## PROGRAM LISTING

```
  10 LET pl=2: LET gl=18: LET hq
c=28: LET dropped=0: LET fired=0
: LET h$="██████": LET h=0: LET
p=0
  20 FOR i=0 TO 7: READ x: POKE
USR "P"+i,x: NEXT i
  30 DATA 0,0,128,252,255,48,96,
0
  40 PAPER 7: BORDER 7: INK 0: C
LS
  50 FOR i=1 TO gl+2: PRINT PAPE
R 5-(i>gl);TAB 31;" ": NEXT i
  60 GO SUB 600
 100 FOR g=1 TO 10
 105 PAPER 8: INK 9: PRINT AT 0,
0;"Raid no.";g
 110 FOR l=gl TO gl+1: GO SUB 65
0: NEXT l: LET l=pl: GO SUB 600
 120 LET pc=0: LET lc=15
 130 LET pc=pc+1: IF pc=30 THEN
GO TO 350
 140 IF NOT dropped THEN IF pc=2
0 OR RND>0.97 THEN LET dropped=1
: LET bl=pl-1: LET bc=pc
 145 PRINT AT pl,pc-1;" P";AT gl
-1,lc;"▄"; OVER 1;AT gl-1,lc;"↑"
 150 IF NOT dropped THEN GO TO 2
50
 160 PRINT AT bl,bc;" ": LET bc=
bc+1: LET bl=bl+2
 170 IF ATTR (bl,bc)<64 THEN PRI
NT AT bl,bc;"o": GO TO 220
```

```
 180 FLASH 1: GO SUB 600: GO SUB
 700: LET h=h+1: LET h$(h)=" ":
FLASH 0: GO SUB 600: LET dropped
=0
 190 IF h<>6 THEN GO TO 350
 200 PRINT AT 20,0;"Your HQ has
been destroyed, but you hit ";p;
" planes"
 210 GO TO 400
 220 IF bl<gl THEN GO TO 250
 230 FLASH 1: PRINT AT bl,bc;"*"
: GO SUB 700: FLASH 0: GO TO 350
 250 IF NOT fired THEN GO TO 300
 260 PRINT AT ml,mc;" ": LET ml=
ml-3: IF ml>pl THEN PRINT AT ml,
mc;"↑": GO TO 300
 270 IF mc=pc THEN GO TO 290
 280 PRINT AT ml,mc; FLASH 1;"*"
: BEEP 1.5,-15: PRINT AT ml,mc;"
": LET fired=0: GO TO 300
 290 PRINT FLASH 1; OVER 1;AT pl
,pc;"*": GO SUB 700: LET p=p+1:
PRINT AT gl+1,12; FLASH 1;"GOT H
IM!": PAUSE 150: GO TO 350
 300 LET n$=INKEY$: IF n$="" THE
N GO TO 130
 310 IF n$="5" THEN IF lc>=2 THE
N PRINT AT gl-1,lc;" ": LET lc=l
c-2: GO TO 130
 320 IF n$="8" THEN IF lc<hqc-3
THEN PRINT AT gl-1,lc;" ": LET l
c=lc+2: GO TO 130
 330 IF n$="0" THEN IF NOT fired
 THEN LET fired=1: LET mc=lc: LE
T ml=gl-2
 340 GO TO 130
 350 INPUT "Press ENTER"; LINE z
$
 355 PRINT AT pl,pc;" ";AT gl-1,
lc;" "
 356 IF fired THEN PRINT AT ml,m
c;" ": LET fired=0
 357 IF dropped THEN LET dropped
=0: IF bl<gl-2 THEN PRINT AT bl,
bc;" "
 360 NEXT g
 380 PRINT AT gl+1,5;"Your HQ ha
s survived. "'"      ";p;" planes
 were shot down"
 400 INPUT "Key 1 to repeat, 0 t
o stop"; LINE z$: IF z$="1" THEN
 RUN
 410 STOP
 600 PRINT BRIGHT 1; PAPER 5; IN
K 1;AT gl-2,hqc-1;h$(1 TO 3);AT
gl-1,hqc-1;h$(4 TO 6): RETURN
 650 PRINT PAPER 8;AT l,0;"
                            ": RE
TURN
```

```
700 FOR i=1 TO 10: BEEP .1,-15:
BEEP .1,-20: NEXT i: RETURN
```

**(Note: The P at line 145 must be entered in graphics mode (G cursor).)**

NOTES

1. Movement. The plane moves one column position at a time; the launcher moves two column positions at a time; the missile moves up three lines at a time; and the bomb moves diagonally one column and two lines at a time.

2. Hitting the HQ. The program tests whether the bomb hits the HQ in a novel way. The HQ is always displayed BRIGHT ly, and this is the only set of positions on the screen with this BRIGHT attribute. Therefore if the bomb is at a position which is BRIGHT then it must have hit the HQ. Using the ATTR function we find that any BRIGHT position will give an ATTR value greater than 64, whereas any non-BRIGHT position will have an ATTR value less than 64. See line 170 in the program listing.

## 6.3 DEAD DUCKS

DESCRIPTION

The player has a double-barrelled shotgun and is situated at the bottom left of the screen. A duck is shown in the right hand half of the screen and the object of the game is to fire up into the air at a specified angle to hit the duck (avoiding the moon also shown). Angles may be between 45 and 85 degrees inclusive. Scores are shown on the screen.

METHOD

```
1.  Initialize variables
    a) Set up moon graphic symbols (A,B,C)
       and duck graphic symbols (D,E,F,G,H,I,J)          (10-80)
    b) Set scores to zero.                                (90)
2.  For go number g from 1 to 5                          (100)
     a) Clear and redraw screen with duck and moon at
        random positions (d,m)                       110-170)
     b) For barrel number b from 1 to 2                   (180)
        (i)  Display barrel number b                      (190)
```

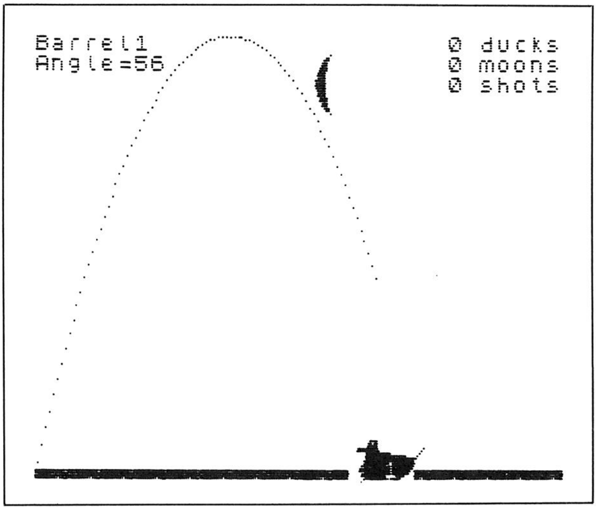
```
          (ii) Enter angle of fire a                           (190)
          (iii) If angle a is invalid then                     (200)
                  A) Message and reenter
                  B) Go to (iii)
          (iv) Display angle a                                 (210)
          (v)  Plot path of bullet. For each bullet
               position (x,y):                             (220-230)
                  A) Plot bullet and sound                 (230-235)
                  B) If bullet hits moon then
                     (I)   Add 1 to moon hits (hm) and display  (260)
                     (II)  Drop moon out of sky with sound      (270)
                     (III) Go to II (c) below                   (275)
                  C) If bullet hits duck
                     (I)  If a glance hit then (a) display
                                               OUCH         (280-290)
                                           (b) sound           (290)
                                           (c) go to (vii)
                      (II) (DIRECT HIT) (a) Display DEAD        (300)
                                        (b) Funeral sound       (300)
                                        (c) Increment and
                                            display shots       (310)
                                        (d) Press
                                            ENTER (310 and SR500)
                                        (e) Go to (c)
                   D) Repeat to (v)                             (242)
          (vi)  Display MISS                                    (245)
          (vii) Increment and display shots s                   (295)
          (viii) Repeat to 2(b)                                 (297)
       c) Repeat to 2                                           (320)
   3.  End message with score                                   (340)
   4.  Stop                                                     (350)
```

VARIABLES

c,d,i = general purpose counters
a$ = user defined graphic (A—-J)
x = binary pattern for user graphic / x-axis position of bullet
hd = number of ducks hit
hm = number of moons hit
s = number of shots
g = go number (1-5)
d = column position of start of duck
m = column position of moon
b = barrel number (1-2)

a = angle of fire
y = y-axis position of bullet
lin, col = line and column position of bullet corresponding to (y,x)
z = counter used to drop moon out of sky
z$ = dummy variable

## SCREEN LAYOUT



```
Barrel1                    0 ducks
Angle=56                   0 moons
                           0 shots
```

## SUBROUTINES

SR500: Press ENTER

## PROGRAM LISTING

```
   10 FOR c=1 TO 10
   20 READ a$
   30 FOR d=0 TO 7
   40 READ x: POKE USR a$+d,x
   50 NEXT d
   60 NEXT c
   70 DATA "A",1,6,14,28,28,60,60
,120,"B",120,120,248,248,248,248
,120,120,"C",120,60,60,28,28,14,
6,1
   80 DATA "D",0,0,0,15,15,0,0,0,
"E",60,108,124,252,254,255,255,2
55,"F",0,0,0,0,0,240,255,255,"G"
```

```
,0,0,0,0,0,0,240,255,"H",0,0,0,8
,16,32,64,128,"I",1,1,3,7,7,3,3,
1,"J",255,255,255,254,252,248,22
4,128
  90 LET hd=0: LET hm=0: LET s=0
 100 FOR g=1 TO 5
 110 CLS : PAPER 1: INK 7: BORDE
R 7: FOR i=0 TO 20: PRINT TAB 31
;" ": NEXT i
 120 LET d=INT (RND*16+12): LET
d2=2*d
 130 LET m=INT (RND*7+12): LET m
2=2*m
 140 PRINT AT 0,0;"Barrel"' FLAS
H 1;"Angle=? "; FLASH 0;AT 0,25;
hd;" ducks";AT 1,25;hm;" moons";
AT 2,25;s;" shots"
 150 PRINT BRIGHT 1; INK 4;AT 1,
m;"A";AT 2,m;"B";AT 3,m;"C"
 160 PRINT INK 3;AT 21,0;"▄▄▄▄▄▄
▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄"
 170 PRINT BRIGHT 1; INK 6;AT 20
,d;"DEFGH";AT 21,d;"I█"; INVERSE
 1;CHR$ (g+48); INVERSE 0;"J"
 180 FOR b=1 TO 2
 190 PRINT AT 0,6;b; FLASH 1;AT
1,0;"Angle=? ": INPUT "Enter ang
le of fire:";a
 200 LET a=INT a: IF a<45 OR a>8
5 THEN INPUT (a;" is not between
 45 and 85."'"Reenter angle:");a
: GO TO 200
 210 PRINT AT 1,0;"Angle=";a
 220 FOR i=0 TO 100
 230 LET x=INT (0.056*i*(90-a)):
 LET f=i*(100-i): LET y=INT (f*0
.07): PLOT x,y
 235 BEEP 0.1,f/100
 240 LET lin=21-INT (y/8): LET c
ol=INT (x/8): IF ATTR (lin,col)>
64 THEN GO TO 250
 242 NEXT i
 245 PRINT AT 20,d-4;"Miss": GO
TO 295
 250 IF lin>3 THEN GO TO 280
 260 LET hm=hm+1: PRINT AT 1,25;
hm
 270 FOR z=1 TO 18: PRINT INK 4;
AT z,m;" ";AT z+1,m;"A";AT z+2,m
;"B";AT z+3,m;"C": BEEP 0.1,1-z:
 NEXT z
 275 GO TO 310
 280 IF col>d AND col<d+4 THEN G
O TO 300
 290 PRINT AT 20,d-4;"Ouch";: BE
EP .5,5: BEEP .5,3
 295 LET s=s+1: PRINT AT 2,25;s
 297 NEXT b
 298 GO TO 320
 300 LET hd=hd+1: PRINT AT 0,25;
hd: PRINT AT 20,d-4;"DEAD": BEEP
 1,-10: BEEP 1,-10: BEEP .5,-10:
 BEEP 1,-10
 310 LET s=s+1: PRINT AT 2,25;s:
 GO SUB 500
 320 NEXT g
 340 PRINT FLASH 1;AT 10,11;"The
 end";AT 12,3;"You killed ";hd;"
 ducks"
 350 STOP
 500 INPUT "Press ENTER to conti
nue ";z$
 510 RETURN
```

**(Note: The characters A,B,C,D,E,F,G,H,I and J variously distributed over lines 150, 170 and 270 must be entered in graphics mode (G cursor).)**

NOTES

1. The path of the bullet is plotted using
x = INT (0.056*i*(90—a)
and y = INT (i*(100—i)*0.07) with i from 1 to 100
The x equation is chosen so that when a=45 then x is almost 255 (i.e. far right of screen) when i is 100. Similarly the y equation is chosen so that the bullet is highest (y approximately = 175) when i is at its midpoint, i.e. 50.

2. The ATTR function is used to detect whether the bullet hits anything useful (i.e. moon or duck) in the same way as in the Headquarters program. Both the moon and the duck are drawn BRIGHTly.

# CHAPTER SEVEN

# DICE GAMES

## 7.1 HIGHER-LOWER

### DESCRIPTION

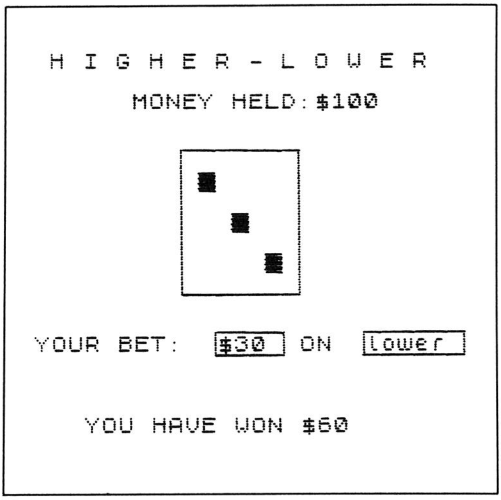
In general, dice games do not require very much skill, apart from awareness of simple probability theory. In Higher-Lower, a die is thrown and the player has to guess whether the next throw is going to produce a higher or lower number. The player places a bet on each attempt, and if the guess is correct the return is double the bet (otherwise the bet is lost). $50 is allocated as starting money and the game ends when either the player runs out of money or reaches $500. In the latter case, the program shows how many attempts it took. The "house" always wins when a die value repeats since the player is not allowed to bet on the same value reappearing.

### METHOD

1. Initialize screen and variables
    - a) Set screen colors (10)
    - b) Print titles (25, 65, 70)
    - c) Set initial money held (m) to 50 and display (30)
    - d) Set up array of die patterns (d$) (40-50)
    - e) Set no. of turns (t) to 0 (55)
    - f) Display die with random value (n) (55-60 and SR600)
2. Player's go
    - a) Set old die value (o) to n (75)
    - b) Clear message lines (SR550)
    - c) Enter bet(b) (80)
    - d) If bet invalid then (90-100)
        - (i) Message and reenter
        - (ii) Go to (d)
    - e) Change money held (m) by −b (110 and SR750)
    - f) Display bet
    - g) Enter whether higher or lower (b$) (140)
    - h) If b$ invalid then (150-155)

```
            (i)  Message and reenter
            (ii) Go to (h)
        i) Display higher/lower                          (160-170)
        j) "Roll" die to random value (n)                (SR600)
        k) Increment number of turns (t)                 (190)
        l) If bet correct then                           (200-215)
          (i)   Message and sound                        (225)
          (ii)  Change money held (m) by 2*b   (220, 230 and SR750)
          (iii) If money equals or exceeds 500 then      (230)
                A) Message and sound
                B) Go to 3
        m) If bet incorrect then                         (240)
            (i)  Message and sound
            (ii) If no money left then
                A) Message and sound
                B) Go to 3
        n) Enter value z$                                (250)
        o) If z$ = 0 then
            (i)  Message showing money left (m)          (260)
            (ii) Go to 3
        p) Go back to 2                                  (250)
    3.  End
        a) Rerun or stop                                 (300-310)
```

SCREEN LAYOUT



```
HIGHER-LOWER
MONEY HELD: $100

YOUR BET: $30 ON Lower

YOU HAVE WON $60
```

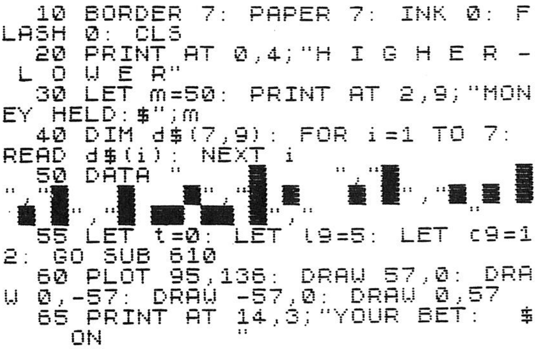
## VARIABLES

m = money held (initially = 50)
d$ = holds die patterns
i, j = general purpose counters
t = number of turns so far
l9, c9 = line and column position at which dice displayed (SR650)
n = current value of die
o = old value of die
b = amount bet
b$ = holds l (lower) or h (higher)
h$ = holds "lower" or "higher", depending on b$
hibet = true if h$ contains "higher"
hidie = true if die is higher than previous (ie. n > o)
incm = amount by which money held is to be changed
z$ = dummy value
m1 = new value of m after incrementing by incm

## ROUTINES

SR550 = clears message areas on screen
SR600 = rolls die to value n. Entered at line 610 when setting up the initial die value at the start. Uses SR650
SR650 = displays die, value n, at position l9, c9
SR750 = changes 'money held' value on screen by incm

## PROGRAM LISTING



```
  10 BORDER 7: PAPER 7: INK 0: F
LASH 0: CLS
  20 PRINT AT 0,4;"H I G H E R -
 L O W E R"
  30 LET m=50: PRINT AT 2,9;"MON
EY HELD:$";m
  40 DIM d$(7,9): FOR i=1 TO 7:
READ d$(i): NEXT i
  50 DATA "    █    ","█       █
",""█   █   █",""█ █   █ █",""█ █ █
 █ █",""█ ██ ██ █"","        "
  55 LET t=0: LET l9=5: LET c9=1
2: GO SUB 610
  60 PLOT 95,136: DRAW 57,0: DRA
W 0,-57: DRAW -57,0: DRAW 0,57
  65 PRINT AT 14,3;"YOUR BET:   $
    ON             "
  70 PLOT 111,64: DRAW 33,0: DRA
W 0,-9: DRAW -33,0: DRAW 0,9: PL
OT 183,64: DRAW 49,0: DRAW 0,-9:
 DRAW -49,0: DRAW 0,9
  75 LET o=n: GO SUB 550
  80 INPUT "Enter amount to bet
:$";b
  90 IF b<1 OR b<>INT b THEN INP
UT "$ notes only - reenter:$";b:
 GO TO 90
 100 IF b>m THEN INPUT ("You don
t have $";b;" - reenter: $");b:
 GO TO 90
 105 IF b=m THEN PRINT AT 16,5;"
An all-or-nothing bet!"
 110 LET incm=-b: GO SUB 750
 120 PRINT AT 14,15;b
 140 INPUT "Higher or lower? (h/
l):"; LINE b$
 150 IF b$<>"h" AND b$<>"l" THEN
 INPUT "Reenter (h/l)"; LINE b$:
 GO TO 150
 155 IF (b$="h" AND o=6) OR (b$=
"l" AND o=1) THEN INPUT "Don't b
e silly!  Reenter (h/l):"; LINE
b$
 160 LET h$="lower ": LET hibet=
0: IF b$="h" THEN LET h$="higher
": LET hibet=1
 170 PRINT AT 14,23; FLASH 1;h$
 180 GO SUB 600
 190 LET t=t+1
 200 LET hidie=(n>o)-(n=o)
 215 IF hidie<>hibet THEN GO TO
240
 220 LET incm=2*b
 225 PRINT AT 18,6;"YOU HAVE WON
 $";incm: BEEP 0.2,20: BEEP 0.5,
5
 230 GO SUB 750: IF m>=500 THEN
PRINT AT 20,0;"You have broken t
he bank in ";t;"   turns": BEEP
1,25: BEEP 1.5,5: GO TO 300
 235 GO TO 250
 240 PRINT AT 18,10;"You are wro
ng": BEEP 1,-15: IF m=0 THEN PRI
NT AT 20,3;"That's it.  No money
 left!": BEEP 1,-5: BEEP 1.5,-15
: GO TO 300
 250 INPUT "Press ENTER for anot
her bet, or 0 to stop"; LINE z$:
 IF z$<>"0" THEN GO TO 75
 260 PRINT AT 20,5;"You stopped
with $";m
 300 INPUT "Press 1 to repeat, 0
 to stop"; LINE z$: IF z$="1" TH
EN RUN
 310 STOP
 550 PRINT AT 14,15;"   ";AT 14,
23;"      "
 560 FOR i=16 TO 18 STEP 2: PRIN
T AT i,0;"
          ": NEXT i
 570 RETURN
 600 INPUT "Press ENTER to roll
die"; LINE z$
 605 LET n=7: GO SUB 650: FOR i=
1 TO 10: BEEP .1,i*2: NEXT i
 610 LET n=INT (RND*6)+1: GO SUB
 650
 620 RETURN
 650 PAPER 2: INK 7
 660 PRINT AT l9,c9;"       "
 670 FOR i=1 TO 3: PRINT AT l9+i
*2-1,c9;" ";: FOR j=3*i-2 TO 3*i
: PRINT d$(n,j);" ";: NEXT j: PR
INT AT l9+i*2,c9;"       ": NEXT
 i
 680 PAPER 7: INK 0: RETURN
 750 LET m1=m+incm
 760 FOR i=m TO m1 STEP 1-2*(m1<
m)
 770 PRINT AT 2,21;i;"  ": BEEP
.01,15
 780 NEXT i
 790 LET m=m1: RETURN
```

**(Note: Enter the contents of the strings at line 50 using GRAPHICS mode with ■ and SPACE—do not use INV. VIDEO as this produces extraneous characters.)**

NOTES

1. Many of the features of Higher-Lower can be directly used in other dice games, and for this purpose several general subroutines are used. In particular, SR650 may easily be transferred to other programs —it draws a 7 x 7 die with its top left hand corner at line 19, column c9, with value n. It uses array d$ and therefore lines 40 and 50 which set up these die patterns would also be needed. The seventh value allows for a blank die.

2. Most dice games are simple in concept, but can be made much more interesting by thoughtful input validation (e.g. line 155, making sure the player does not bet "Higher" when the die shows a 6); by suitable messages (e.g. line 105); and by good screen layout.

3. You must use lower case for this game (as well as for the other two games in this chapter).

## 7.2 TRIPLE

### DESCRIPTION

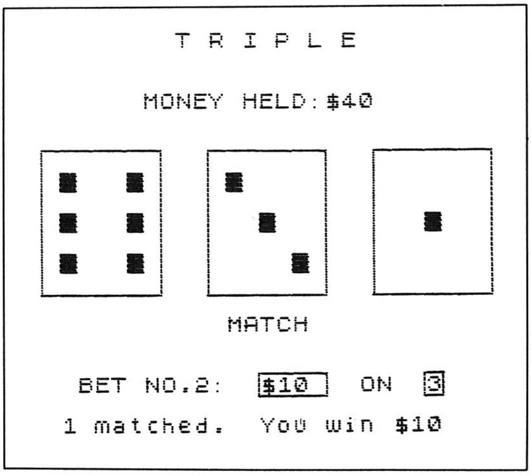
The player starts with $50 and bets on the occurrence of a number on three dice. If the chosen number comes up on one die, the player recovers the bet, if it appears on two dice, the player wins at two-to-one odds, and if it is on all three dice, the win is at three-to-one. The game ends when the player runs out of money or if the win is more than $500.

### METHOD

```
1.  Initialize screen and variables
    a) Screen colors                                      (10)
    b) Print titles                                   (20, 80)
    c) Set money (m) to 50 and display                    (30)
    d) Set up array of dice patterns (d$)              (40-50)
    e) Draw dice shapes                            (70,90,100)
    f) Set turns (t) to 0                                (110)
2.  Player's go
    a) Clear message lines                               (120)
    b) Display dice blank                                (130)
    c) Add 1 to turns (t) and display                    (140)
    d) Set matches (ms) to 0                             (140)
    e) Enter bet amount (b)                              (150)
    f) If b invalid then                             (160-170)
       (i) Message and reenter
       (ii) Go to (f)
    g) Display bet (b)                                   (180)
    h) Change money held (m) by -b            (180 and SR700)
    i) Enter bet number (bn)                             (190)
    j) If bn invalid then                                (200)
       (i) Message and reenter
       (ii) Go to (j)
    k) Display bet number (bn)                           (210)
    l) 'Roll' dice to values n(1),n(2),n(3)   (220 and SR500)
    m) For each die                                      (230)
       (i) If die = bet number (bn) then                 (240)
             A) Add 1 to matches (ms)
             B) Message (MATCH)
       (ii) Repeat to (m)                                (260)
    n) If no matches (ms=0) then                         (270)
```

```
            (i)  Message (NO MATCHES)
            (ii) If no money left (m=0) then
                  A) Message (OUT OF MONEY)
                  B) Go to (3)
            (iii) Go to (p)                                   (275)
         o) (Matches)
            (i)  Message showing winnings                     (280)
            (ii) Change money held (m) by b*ms              (SR700)
            (iii) If money (m) equals or exceeds $500 then    (290)
                  A) Message (BROKE BANK)
                  B) Go to (3)
         p) Enter value z$                                    (300)
         q) If z$ = 0 then
            (i)  Message showing money left (m)               (310)
            (ii) Go to (3)
         r) Go back to (2)                                    (300)
      3. End
         a) Rerun or stop                                 (320-330)
```

SCREEN LAYOUT



```
T R I P L E

MONEY HELD:$40

MATCH

BET NO.2: $10 ON 3
1 matched. You win $10
```

VARIABLES

m = money held
d$ = holds dice patterns
i,j,k = general purpose counter

array n = holds values of three dice
t = number of turns so far
n = value of dice displayed by SR500
b = amount bet
incm = amount by which money held is to be changed
bn = number bet upon
z$ = dummy value
ms = number of matches between bn and array d
l9, c9 = line and column position at which a die is displayed (SR550)
m1 = new value of m after incrementing by incm

ROUTINES

SR500 = calls SR550 three times, with dice no. = 1,2,3 respectively
SR550 displays dice with random value, or set value (n) if n not equal to 0
SR600 = draws edge of dice
SR700 = changes value of money held by incm

PROGRAM LISTING

```
  10 INK 0: PAPER 7: BORDER 7: F
LASH 0: CLS
  20 PRINT AT 0,11;"T R I P L E"
  30 LET m=50: PRINT AT 3,9;"MON
EY HELD:$";m
  40 DIM d$(7,9): FOR i=1 TO 7:
READ d$(i): NEXT i
  50 DATA "    █    ","█       █
","█   █   █","█ █   █ █","█ █ █
 █ █","█ ██ ██ █","         "
  60 DIM n(3)
  70 PLOT 23,128: GO SUB 600: PL
OT 103,128: GO SUB 600: PLOT 183
,128: GO SUB 600
  80 PRINT AT 17,5;"BET NO.    $
     ON"
  90 PLOT 127,40: DRAW 33,0: DRA
W 0,-9: DRAW -33,0: DRAW 0,9
 100 PLOT 207,40: DRAW 9,0: DRAW
 0,-9: DRAW -9,0: DRAW 0,9
 110 LET t=0
 120 PRINT AT 17,17;"   ";AT 17,
26;" ";AT 14,0;"
                ";AT 19,0;"
                           "

 130 LET n=7: GO SUB 500
 140 LET t=t+1: PRINT AT 17,12;t
;":": LET ms=0
```

```
 150 INPUT "Enter amount to bet:
 $";b
 160 IF b<1 OR b<>INT b THEN INP
UT "$ notes only - reenter: $";b
: GO TO 160
 170 IF b>m THEN INPUT ("You don
't have $";b;" - reenter: $");b:
 GO TO 160
 180 PRINT AT 17,17;b: LET incm=
-b: GO SUB 700
 190 INPUT "Enter number on whic
h to bet: ";bn
 200 IF bn<>INT bn OR bn<1 OR bn
>6 THEN INPUT "No such number -
reenter: ";bn: GO TO 200
 210 PRINT AT 17,26; FLASH 1; IN
K 3;bn
 220 INPUT "Press ENTER to roll
dice"; LINE z$: LET n=0: GO SUB
500
 230 FOR i=1 TO 3
 240 IF n(i)=bn THEN LET ms=ms+1
: PRINT FLASH 1;AT 14,10*i-6;"MA
TCH": BEEP 1,20: BEEP 1,10: GO T
O 260
 260 NEXT i
 265 IF ms>0 THEN GO TO 280
 270 PRINT AT 19,11;"No matches"
: IF m=0 THEN PRINT AT 20,3; INK
 1;"That's it!  No money left!":
 GO TO 320
 275 GO TO 300
 280 PRINT AT 19,4;ms;" matched.
  You win $";b*ms: LET incm=b*ms
: GO SUB 700
 290 IF m>=500 THEN PRINT AT 20,
0; INK 1;"You have broken the ba
nk in ";t;"  turns": GO TO 320
 300 PAUSE 100: INPUT "Press ENT
ER for another bet or  0 to stop
"; LINE z$: IF z$<>"0" THEN GO T
O 120
 310 PRINT AT 20,5; INK 1;"You s
topped with $";m
 320 PAUSE 150: INPUT "Press 1 t
o repeat, 0 to stop"; LINE z$: I
F z$="1" THEN RUN
 330 STOP
 500 LET l9=6: LET c9=3: LET i=1
: GO SUB 550
 510 LET i=2: LET c9=13: GO SUB
550
 520 LET i=3: LET c9=23: GO SUB
550
 530 RETURN
 550 LET n(i)=INT (RND*6)+1
 560 IF n<>0 THEN LET n(i)=n
 570 PAPER 2: INK 7
 580 PRINT AT l9,c9;"       "
 590 FOR k=1 TO 3: PRINT AT l9+k
*2-1,c9;" ";: FOR j=3*k-2 TO 3*k
: PRINT d$(n(i),j);" ";: NEXT j:
 PRINT AT l9+k*2,c9;"       ": N
EXT k
 592 IF n=0 THEN BEEP .5,10
 595 PAPER 7: INK 0: RETURN
 600 DRAW 57,0: DRAW 0,-57: DRAW
 -57,0: DRAW 0,57: RETURN
 700 LET m1=m+incm
 710 FOR i=m TO m1 STEP 1-2*(m1<
m)
 720 PRINT AT 3,21;i;"  ": BEEP
.01,15
 730 NEXT i
 740 LET m=m1: RETURN
```

**(Note: Enter the contents of the strings at line 50 using GRAPHICS mode with ■ and SPACE—do not use INV. VIDEO as this produces extraneous characters.)**

NOTES

1. Subroutine 550 is used in two ways in the program: to generate and display a random dice value, or to display a blank die, for example at the start of each go. Value n is used to indicate which is required: at line 220 the routine is called (via SR500) with n = 0, and n = 7 when SR500 is called at line 130 and SR550 then displays the dice pattern corresponding to 7 in the DATA statements of line 50 i.e. a blank die is displayed.

## 7.3 PLAYING THE NUMBERS

DESCRIPTION

In this game two dice are used and the player and the TS2068 are opponents. On each attempt the player (or TS2068) has to guess three features of the two dice to be thrown, namely whether the sum will be odd, what the sum will be, and what the dice values will be. Correct answers to these three questions give scores of one, two, and three respectively. The first player to reach eleven wins. The TS2068 does not cheat by looking ahead, and just to be generous, the user always starts first.

METHOD

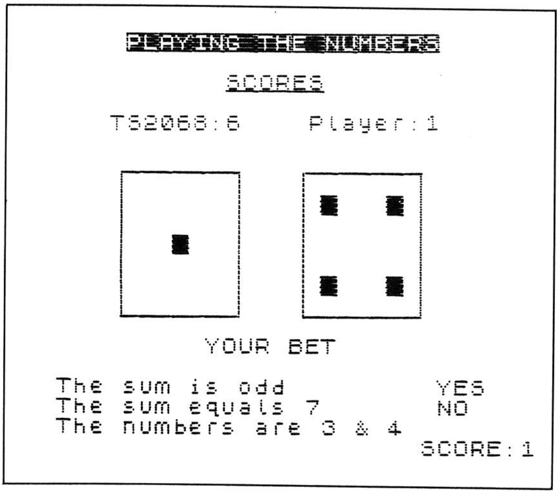
1. Initialize screen and variables
   a) Set screen colors (10)

```
   b) Set up array of dice pattern d$                     (40-50)
   c) Print titles and draw dice                          (60-95)
2. Player's go
   a) Set players go to 1                                   (100)
   b) Enter whether odd or even                             (110)
   c) If invalid then                                       (120)
      (i)  Message and reenter
      (ii) Go to c
   d) Display odd/even                                  (130-140)
   e) Enter sum of dice (sb)                                (150)
   f) If invalid or inconsistent then                   (160-180)
      (i)  Message and reenter
      (ii) Go to f
   g) Display sum sb                                        (190)
   h) Enter dice numbers b1,b2                              (200)
   i) If invalid or inconsistent then                   (210-220)
      (i)  Message and reenter
      (ii) Go to i
   j) Display dice numbers b1,b2                            (230)
   k) 'Roll' the dice, giving n1 and n2                   (SR500)
   l) Score the bet against n1 and n2                     (SR700)
   m) Add score to player's score, ps                       (250)
   n) If ps greater than or equal to 11 then                (260)
      (i)  Message declaring player the winner
      (ii) Go to 4
   o) Press ENTER and clear screen                        (SR850)

3. TS2068's go
   a) Set players go to 0                                   (270)
   b) Generate numbers b1 and b2 at random                  (280)
   c) Calculate sum of b1 and b2 (sb)                       (290)
   d) Display TS2068's bet: whether sb odd even, sb itself
      and b1 and b2                                     (290-310)
   e) Roll the dice giving n1 and n2                      (SR500)
   f) Score the bet against n1 and n2                     (SR700)
   g) Add score to TS2068's score, ss                       (330)
   h) If ss greater than or equal to 11 then                (340)
      (i)  Message declaring TS2068 the winner
      (ii) Go to 4
   i) Press ENTER and clear screen                        (SR850)
   j) Go back to (2)                                        (350)

4. End
   a) Sound                                                 (360)
   b) Rerun or stop                                     (380-390)
```

## SCREEN LAYOUT



```
          PLAYING THE NUMBERS

                 SCORES

         TS2068:6      Player:1

              YOUR BET

  The sum is odd              YES
  The sum equals 7            NO
  The numbers are 3 & 4
                            SCORE:1
```

## VARIABLES

d$ = holds dice patterns
i,j = general purpose counters
n1 = value displayed on first die
n2 = value displayed on second die
ps = player's score
ss = TS2068's score
x$ = "o"/"e" - player's entry of odd/even
evenbet = true if bet is even
evencalc = true if sum of two numbers chosen by player is even
evensum = true if dice give even sum
e$ = "even"/"odd"
sb = sum bet
b1, b2 = dice numbers bet
score = score attained by one bet
playersgo = true when it is the player's go
z$ = dummy value
n = dice value parameter used by SR650
l9, c9 = line of column position at which dice displayed by SR650
l = line position
c = column position

## ROUTINES

SR450 = draws lines around the edges of dice
SR500 = rolls dice to numbers n1 and n2. Uses SR 600
SR600 = sets positions to display dice and calls SR650
SR650 = displays a die, value n, at position l9, c9
SR700 = scores a bet
SR850 = clears bet lines and dice

## PROGRAM LISTING

```
  10 INK 0: PAPER 7: BORDER 7: F
LASH 0: BRIGHT 0: CLS
  20 FOR i=0 TO 7: READ x: POKE
USR "G"+i,x: NEXT i
  30 DATA 170,85,170,85,170,85,1
70,85
  40 DIM d$(7,9): FOR i=1 TO 7:
READ d$(i): NEXT i
  50 DATA "    ▒    ","▒       ▒
","▒   ▒   ▒","▒ ▒   ▒ ▒","▒ ▒ ▒
 ▒ ▒","▒ ▒▒ ▒▒ ▒","         "
  60 PRINT AT 0,6; INVERSE 1;"PL
AYING THE NUMBERS"; INVERSE 0;AT
 2,12;"SCORES";AT 2,12; OVER 1;"
______"; OVER 0;AT 4,7;"TS2068:0
   Player:0"
  70 PLOT 55,120: GO SUB 450
  80 PLOT 143,120: GO SUB 450
  90 LET n1=7: LET n2=7:: GO SUB
 600: LET ps=0: LET ss=0: DEF FN
 d()=INT (RND*6)+1
  95 GO SUB 850
 100 PRINT AT 15,12;"YOUR BET":
LET playersgo=1
 110 INPUT "Odd or even sum? (o/
e)";x$
 120 IF x$<>"o" AND x$<>"e" THEN
 INPUT "Reenter (o/e)";x$: GO TO
 120
 130 LET evenbet=0: LET e$="odd"
: IF x$="e" THEN LET evenbet=1:
LET e$="even"
 140 PRINT AT 17,3;"The sum is "
;e$
 150 INPUT "Sum=";sb
 160 IF sb<>INT sb OR sb<2 OR sb
>12 THEN INPUT "Impossible.  Ree
nter (2-12):";sb: GO TO 160
 170 LET evencalc=(sb=2*INT (sb/
2))
 180 IF evenbet<>evencalc THEN I
NPUT (sb;" is not ";e$;".  Reent
er:");sb: GO TO 160
 190 PRINT AT 18,3;"The sum equa
ls ";sb
 200 INPUT "Enter dice nos:(1)="
;b1;"  (2)=";b2
```

```
 210 IF b1<>INT b1 OR b2<>INT b2
 OR b1<1 OR b2<1 OR b1>6 OR b2>6
 THEN INPUT "Impossible.  Reente
r(1)=";b1;" (2)=";b2: GO TO 210
 220 IF b1+b2<>sb THEN INPUT (b1
;"+";b2;"<>";sb;"  Reenter (1)="
);b1;" (2)=";b2: GO TO 210
 230 PRINT AT 19,3;"The numbers 
are ";b1;" & ";b2
 240 GO SUB 500: GO SUB 700
 250 LET ps=ps+score
 260 IF ps>=11 THEN PRINT AT 4,1
4; FLASH 1;ps;AT 21,7;"You are t
he winner!": GO TO 360
 265 GO SUB 850
 270 PRINT AT 15,13;"MY BET": LE
T playersgo=0
 280 LET b1=FN d(): LET b2=FN d(
)
 290 LET sb=b1+b2: LET evenbet=(
sb=2*INT (sb/2))
 300 LET e$="odd": IF evenbet TH
EN LET e$="even"
 310 PRINT AT 17,3;"The sum is "
;e$;AT 18,3;"The sum equals ";sb
;AT 19,3;"The numbers are ";b1;"
 & ";b2
 320 GO SUB 500: GO SUB 700
 330 LET ss=ss+score
 340 IF ss>=11 THEN PRINT AT 4,1
4; FLASH 1;ss;AT 21,8;"I am the 
winner": GO TO 360
 350 GO TO 95
 360 BEEP .5,20: BEEP .5,10: BEE
P .5,0
 370 PAUSE 250
 380 INPUT "Enter 1 to repeat, 0
 to stop"; LINE z$: IF z$="1" TH
EN RUN
 390 STOP
 450 DRAW 57,0: DRAW 0,-57: DRAW
 -57,0: DRAW 0,57: RETURN
 500 INPUT "Press ENTER to roll 
dice"; LINE z$
 510 LET n1=FN d(): LET n2=FN d(
): GO SUB 600
 520 RETURN
 600 LET n=n1: LET l9=7: LET c9=
7: GO SUB 650
 610 LET n=n2: LET c9=18: GO SUB
 650
 620 RETURN
 650 PAPER 2: INK 7
 660 PRINT AT l9,c9;"       "
 670 FOR i=1 TO 3: PRINT AT l9+i
*2-1,c9;" ";: FOR j=3*i-2 TO 3*i
: PRINT d$(n,j);" ";: NEXT j: PR
INT AT l9+i*2,c9;"       ": NEXT
 i
 680 PAPER 7: INK 0: RETURN
 700 LET sd=n1+n2: LET score=0
 710 LET evensum=(sd=2*INT (sd/2
))
 720 IF evensum<>evenbet THEN PR
INT AT 17,26; INK 2;"NO": GO TO
780
 730 LET score=1: PRINT AT 17,26
; INK 4;"YES"
 740 IF sd<>sb THEN PRINT AT 18,
26; INK 2;"NO": GO TO 780
 750 LET score=score+2: PRINT AT
 18,26; INK 4;"YES"
 760 IF (n1=b1 AND n2=b2) OR (n1
=b2 AND n2=b1) THEN PRINT AT 19,
26; INK 4;"YES": LET score=score
+3: GO TO 780
 770 PRINT AT 19,26; INK 2;"NO"
 780 PRINT AT 20,25;"SCORE:";sco
re
 790 LET l=4: LET c=14:: LET s=s
s: IF playersgo THEN LET c=25: L
ET s=ps
 800 FOR i=s TO s+score: PRINT A
T l,c;i: BEEP .1,20: NEXT i
 810 RETURN
 850 INPUT "Press ENTER:"; LINE
z$: FOR l=15 TO 20: PRINT AT l,0
;"
  ": NEXT l
 860 LET l9=7: LET c9=7: LET n=7
: GO SUB 650: LET c9=18: GO SUB
650
 870 RETURN
```

**(Note: Enter the contents of the strings at line 50 using GRAPHICS mode with ■ and SPACE—do not use INV. VIDEO as this produces extraneous characters.)**

NOTES

1. Much of the validation built in to the ''player's go' section of the program is necessary because of the order in which the three questions about the dice are asked. It could be made much simpler if just the two dice numbers were asked for and the program calculated whether the sum was odd or even, and what the sum was, as it does for its own turn. However, without validation some rather silly responses are possible.

2. The use of several logical variables, such as evensum and playersgo, may seem longwinded but they do aid the clarity and readibility of what can be a somewhat tortuous set of coding.

# CHAPTER EIGHT

# CARD GAMES

## 8.1 INBETWEEN

### DESCRIPTION

Twenty cards are shown on the screen, face down. The first two cards are then revealed, and the player then has to bet whether or not the third card's value will lie in between the values of the first two cards. If correct, the player wins at 2 to 1 odds, otherwise the bet is lost. The game continues with betting whether card 4 will be between card 3 and card 2, and so on.

Note that the suits of the cards are irrelevant, they are shown for realism but do not contribute to the game. Ace is low and a card is considered to lie between two other cards if its value is in the **inclusive** range. Ten is shown as the letter T in this and subsequent card games.

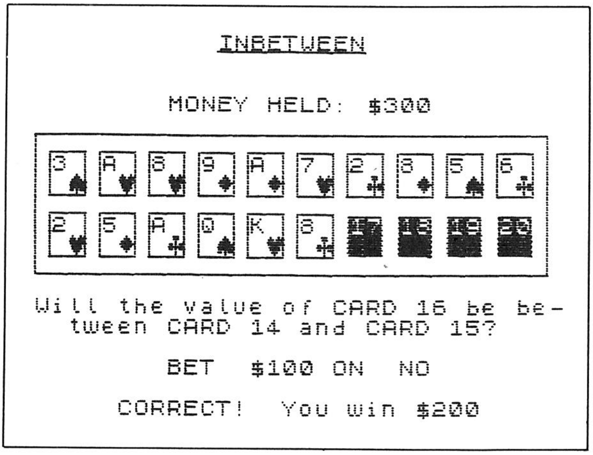
The player starts with $50 and the game ends when the player either exceeds $500 or has no money left.

### METHOD

1. Initialize screen and variables
   a) Set screen colors (10)
   b) Read graphics for hearts, clubs, diamonds and spades (20-30)
   c) Initialize s$ to contain suit symbols and v$ values (40)
   d) Put suit graphics into array s (60)
   e) Put card numbers 1-52 into array c (70)
   f) Shuffle array c (80)
   g) Set corresponding card values in array v and suits in array t (85)
   h) Print titles (100)
   i) Set money held (m) to 50, and display (100)

```
     j) Display twenty cards, face down                    (105-135)
     k) Set third card (cd3) to 3                              (140)
     l) Reveal first two cards                     (150 and SR500)
 2.  Player's go
     a) Clear message lines                                    (190)
     b) Enter whether inbetween (y/n)                      (200-210)
     c) If invalid entry then                                  (220)
        (i)  Message and reenter
        (ii) Go to (c)
     d) Enter amount of bet (b)                                (240)
     e) If invalid then                                    (250,260)
        (i)  Message and reenter
        (ii) Go to (e)
     f) Display bet                                            (280)
     g) Decrease money held by bet                 (290 and SR650)
     h) Press ENTER                                            (300)
     i) Display card numbers cd3 face up
                                          (310 and SR600 and 500)
     j) Calculate whether card's value (v(cd3)) lies       (320-340)
       between values of two previous cards, cd3-1 and cd3-2
     k) If result corresponds with bet then                    (350)
        (i)   Congratulatory message and sound; display
              winnings                                         (360)
        (ii)  Add amount 2*b to money held         (370 and SR650)
        (iii) If money m equals or exceeds $500 then           (380)
                 A) Message BROKEN BANK
                 B) Go to 3
        (iv) Go to (m)                                         (390)
     l) (WRONG)
        (i)  Message NO MORE CARDS and show money left         (400)
        (ii) Go to 3                                           (410)
                 A) Message NO MONEY LEFT
                 B) Go to 3
     m) Add 1 to card counter cd3                              (420)
     n) If counter cd3 exceeds 20 then                         (420)
        (i)  Message NO MORE CARDS and show money left
        (ii) Go to 3
     o) Enter value z$                                         (425)
     p) If z$ not equal to 0 then go back to 2
     q) Message showing money left (m)                         (430)
 3.  End
     a) Rerun or stop                                      (440-450)
```

## SCREEN LAYOUT



```
                INBETWEEN

           MONEY HELD:  $300

Will the value of CARD 16 be be-
   tween CARD 14 and CARD 15?

           BET  $100  ON  NO

       CORRECT!  You win $200
```

## VARIABLES

i,j = general purpose counters
x$ = graphic character being initialized (suit symbol)
x = binary pattern used for setting up graphic
s$ = suit initials H, C, D, S
v$ = values of cards A, 2, 3, ..., 9, T, J, Q, K
array c = holds the numbers 1-52 representing individual cards
array t = holds the suit numbers of the cards in array c
array v = holds the value numbers of the cards in array c
array s = holds the codes of the graphic characters for the suits
r = a random number (1-52) used in shuffling the cards
z,z$ = dummy values
m = money held
cd3 = counter showing position of card currently bet upon (3-20)
a$ = user's answer to question "Inbetween?"
betin = true if user's bets on inbetween
b = amount of bet in $
w$ = YES/NO corresponding to a$
incm = amount by which money held is to be changed (by SR650)
n = parameter to SR600 (card sequence number)
max = larger value of the two cards preceding card cd3

min = smaller value of the two cards preceding card cd3
in = true if the card does lie between the two preceding cards
vn = holds value of the card currently being displayed by SR500
sn = holds suit number of card currently being displayed by SR500
l1, c1 = line and column position of card number n on screen
x1 = units part of number n
x10 = tens part of number n
m1 = new value of m after being changed by incm

## SUBROUTINES AND FUNCTIONS

SR500 displays a card, sequence number n, at positions (l1,c1)
SR600 calculates screen position (l1,c1) of card, sequence number n
SR650 changes displayed value of money left by incm

FN b = calculates larger value of two numbers, a and b
FN s = calculates smaller value of two numbers, a and b

## PROGRAM LISTING

```
  10 PAPER 7: INK 0: FLASH 0: BO
RDER 7: CLS
  20 FOR i=1 TO 4: READ x$: FOR
j=0 TO 7: READ x: POKE USR x$+j,
x: NEXT j: NEXT i
  30 DATA "H",34,119,127,127,62,
62,28,8,"C",28,28,8,107,127,107,
8,8,"D",8,28,62,127,62,28,8,0,"S
",8,28,62,127,127,127,107,8
  40 LET s$="HCDS": LET v$="A234
56789TJQK"
  50 DIM c(52): DIM t(52): DIM v
(52): DIM s(4)
  60 FOR i=1 TO 4: LET s(i)=CODE
 s$(i)+79: NEXT i
  70 FOR i=1 TO 52: LET c(i)=i:
NEXT i
  80 FOR i=1 TO 52: LET r=INT (5
2*RND)+1: LET z=c(i): LET c(i)=c
(r): LET c(r)=z: NEXT i
  85 FOR i=1 TO 52: LET t(i)=INT
 ((c(i)-1)/13)+1: LET v(i)=c(i)-
13*(t(i)-1): NEXT i
  90 DEF FN b(a,b)=a*(a>b)+b*(a<
=b): DEF FN s(a,b)=a+b-FN b(a,b)
 100 LET m=50: PRINT AT 0,11; IN
K 1;"INBETWEEN"; OVER 1;AT 0,11;
"__________"; INK 0; OVER 0;AT 3,
8;"MONEY HELD: $";m
```

```
 105 FOR i=5 TO 11: PRINT PAPER
4;AT i,0;TAB 31: NEXT i
 110 PRINT AT 6,0;: GO SUB 550:
PRINT : GO SUB 550
 120 FOR i=1 TO 10: PRINT AT 6,3
*i-2; INVERSE 1;i: NEXT i
 130 FOR i=11 TO 20: PRINT AT 9,
i*3-32; INVERSE 1;i: NEXT i
 135 PLOT 0,135: DRAW 247,0: DRA
W 0,-55: DRAW -247,0: DRAW 0,55
 140 LET cd3=3
 150 PAUSE 200: LET l1=6: LET c1
=1: LET n=1: GO SUB 500: LET c1=
4: LET n=2: GO SUB 500
 190 FOR i=13 TO 18: PRINT AT i,
0;"
   ": NEXT i
 200 PRINT AT 13,0;"Will the val
ue of CARD ";cd3;" be be-  tween
 CARD ";cd3-2;" and CARD ";cd3-1
;"?"
 210 INPUT "Inbetween? (y/n)"; L
INE a$
 220 IF a$<>"y" AND a$<>"n" THEN
 GO TO 210
 230 LET betin=(a$="y")
 240 INPUT "How much will you be
t? $";b
 250 IF b<>INT b OR b<1 THEN INP
UT "$ notes only.  How much? $";
b: GO TO 250
 260 IF b>m THEN INPUT ("You don
't have $";b;" - reenter $");b:
GO TO 250
 270 LET w$="NO": IF betin THEN
LET w$="YES"
 280 PRINT AT 16,8; INK 3;"BET
$";b;" ON  "; FLASH 1;w$
 290 LET incm=-b: GO SUB 650
 300 INPUT "Press ENTER to turn
over card";z$
 310 LET n=cd3: GO SUB 600: GO S
UB 500
 320 LET max=FN b(v(cd3-1),v(cd3
-2))
 330 LET min=FN s(v(cd3-1),v(cd3
-2))
 340 LET in=((v(cd3)>=min) AND (
v(cd3)<=max))
 350 IF NOT (in=betin) THEN GO T
O 400
 360 BEEP .5,25: BEEP .5,15: PRI
NT AT 18,5; INK 2;"CORRECT!  You
 win $";2*b
 370 LET incm=2*b: GO SUB 650
 380 IF m>=500 THEN PRINT AT 20,
0;"You have broken the bank in "
;4-cd3;"    turns": GO TO 440
```

```
 390 GO TO 420
 400 BEEP 1,-15: PRINT AT 18,9;"
You are wrong"
 410 IF m=0 THEN PRINT AT 20,3;
INK 1;"That's it!  No money left
!": GO TO 440
 420 PAUSE 100: LET cd3=cd3+1: I
F cd3>20 THEN PRINT AT 20,3; INK
 2;"No more cards - you have $";
m: GO TO 440
 425 INPUT "Press ENTER for anot
her bet or 0 to stop"; LINE z$:
IF z$<>"0" THEN GO TO 190
 430 PRINT AT 20,5; INK 1;"You s
topped with $";m
 440 PAUSE 100: INPUT "Press 1 t
o repeat, 0 to stop"; LINE z$: I
F z$="1" THEN RUN
 450 STOP
 500 PAPER 7: LET vn=v(n): LET s
n=t(n): INK 2-(sn=2*INT (sn/2))*
2
 510 PRINT AT l1,c1;v$(vn);" ";A
T l1+1,c1;" ";CHR$ s(sn)
 520 INK 0
 530 PLOT c1*8-1,176-8*l1: DRAW
17,0: DRAW 0,-17: DRAW -17,0: DR
AW 0,17
 540 BEEP 1,5: RETURN
 550 FOR i=1 TO 2: PRINT PAPER 4
;" ██ ██ ██ ██ ██ ██ ██ ██ ██ ██
 ": NEXT i: RETURN
 600 LET x10=INT (n/10): LET x1=
n-10*x10
 605 IF x1=0 THEN LET x10=x10-1:
 LET x1=10
 610 LET l1=6+3*x10
 620 LET c1=3*x1-2
 630 RETURN
 650 LET m1=m+incm
 660 FOR i=m TO m1 STEP 1-2*(m1<
m)
 670 PRINT AT 3,21;i;"  ": BEEP
.01,15
 680 NEXT i
 690 LET m=m1: RETURN
```

NOTES

1. Many of the instructions and routines included above may be used in any card game program. Consider first the method by which the shuffled set of cards is set up. Array c holds 52 values of 1-52 where 1 represents the ace of hearts, 2 represents the two of hearts and so on up to 52 representing the king of spades. Thus suits are organized in hearts, clubs, diamonds, spades sequence and within a suit the values start at ace and go up to king. The card numbers are

initially put into array c in sequence (line 70 above) and are then "shuffled." There are several techniques of doing this, but the simplest is shown at line 80: each card in the 'pack' is swapped with another card in a random position.

2. It is not sufficient to display a card as just a number between 1 and 52 so it is next considered how this number, e.g. 40, is converted to a recognizable form, e.g. A S for the ace of spades. A card has a specific position number or value, and a suit. Its suit is easily calculated since

hearts = numbers 1-13
clubs = numbers 14-26
diamonds = numbers 27-39
spades = numbers 40-52

i.e. given a card number c, its suit t can be found from the formula:

t = INT ((c—1)/13)+1

This is used in line 85 of Inbetween. Similarly the position or value of a card c in suit t is found by

v = c—13*(t—1)

also used in line 85 above.

The variables v and t can then each be used as subscripts of a string:

Take v$ as "A23456789TJQK" (the values)
and s$ as "HCDS" (the suits)

and then the value of a card c is v$(v) and its suit is s$(t). It is finally a simple matter to use the corresponding graphic characters of H, C, D and S for added realism.
For ease of use, the program calculates an array of values numbers (v) and an array of suit numbers (t) corresponding to the card array (c) so that card c(i) has value v(i) and suit t(i).

3. Subroutine 500 is sufficiently generalized to be used in other card game programs—it displays a 2x2 card at a position with its top left hand corner at line l1, column c1. It follows the normal conventions to display hearts and diamonds in red, and clubs and spades in black, with a white background for each.

## 8.2 CONCENTRATION

### DESCRIPTION

This is the familiar game of memory. Fifty two cards are laid out face down in six rows. The player has to pick a pair of cards (by specifying the card sequence numbers shown on the back), the cards are turned over and if they have the same value (e.g. Ace or 5) then they remain turned face up. If they are not of the same value they are turned face down again. The game continues until all the cards are showing face up. Clearly the player's memory has to be exercised in visualizing the values of cards previously turned up.

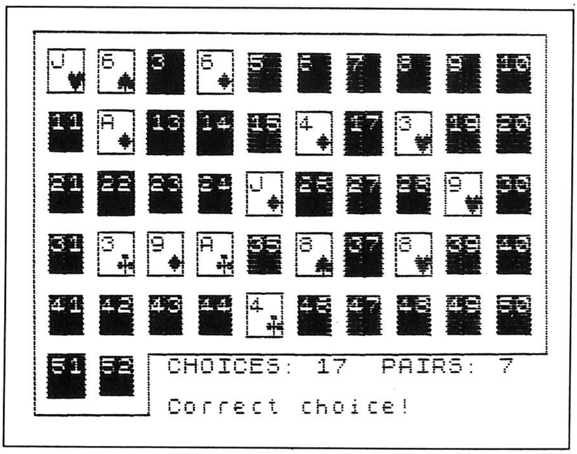
METHOD

```
1.  Initialize screen and variables
    a) Set screen colors and clear screen                        (10)
    b) Set up graphics for card suits                         (20-30)
    c) Initialize suit (s$) and value (v$) symbols               (40)
    d) Put suit graphics into array s                            (60)
    e) Put card numbers 1-52 into array c                        (70)
    f) Shuffle array c                                           (80)
    g) Set corresponding card values and suits in arrays
       v and t                                                   (90)
    h) Display numbered backs of cards on table              (120-150)
    i) Set choices (ch) and pairs (pr) to 0 and display         (160)
2.  Player's choice
    a) Clear message line                                       (200)
    b) Enter first member of pair (p1)                          (200)
    c) If entry invalid then
       (i)  Message and reenter                             (210-220)
       (ii) Go to (c)
    d) Turn over card p1                (230 and SR's600 and 500)
    e) Enter second member of pair                              (240)
    f) If entry invalid then                                (250-270)
       (i)  Message and reenter
       (ii) Go to (f)
    g) Turn over card p2                (280 and SR's600 and 500)
    h) Add 1 to number of choices (ch) and display              (290)
    i) If value of card p1 = value of card P2 then              (300)
       (i)   Message CORRECT and sound                          (310)
       (ii)  Mark cards p1 and p2 as chosen (set values to 0)
       (iii) Add 1 to number of pairs (pr) and display
       (iv)  If pairs (pr) = 26 then                            (320)
                 A) Message FINISHED and sound
                 B) Go to 3
       (v)   Go to (k)                                         (330 )
    j) (NO MATCH)                                               (340)
       (i)   Message NO MATCH and sound
       (ii)  Wait two seconds
       (iii) Turn cards p1 and p2 face down                   (SR800)
```

k) Press ENTER (350)
l) Go to 2

3. End
a) Return or stop (360-380)

SCREEN LAYOUT



```
CHOICES: 17  PAIRS: 7

Correct choice!
```

VARIABLES

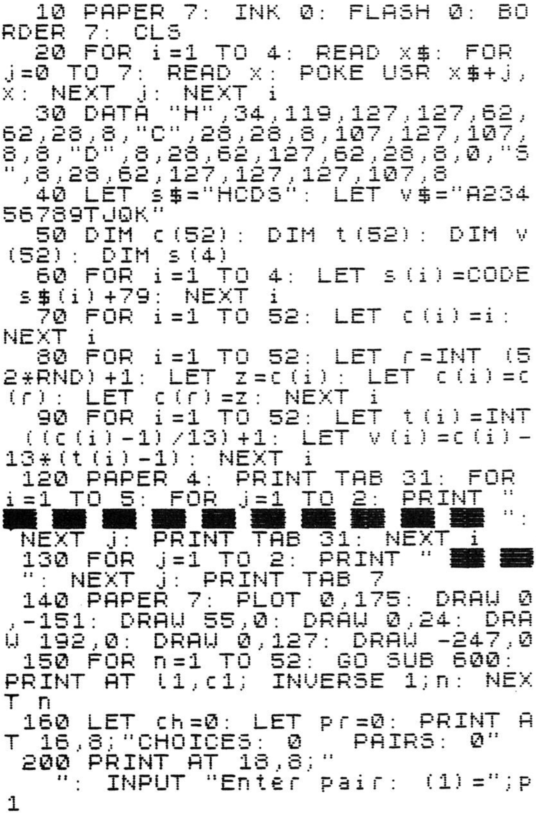
i,j = general purpose counters
x$ = graphic character
x = binary pattern for graphic character
s$ = suit initials, H,C,D,S
v$ = values of cards A,2,3,...,9,T,J,Q,K
array c = holds the numbers 1-52 representing individual cards
array t = holds the suit numbers of the cards in array c
array v = holds the value numbers of the cards in array c
array s = holds the codes of the graphic characters for the suits
r = a random number (1-52) used in shuffling the cards
z,s$ = dummy values
ch = number of choices (of pairs) made
pr = number of correct pairs chosen
p1 = first member of a pair, as a card sequence number
p2 = second member of a pair
n = parameter to SR600 - a card sequence number
vn = value number of a card (in SR500)
sn = suit number of a card (in SR500)

x1 = units part of number n
x10 = tens part of a number n
l1,c1 = line and column position of a card

SUBROUTINES

SR500 displays a card, sequence number n, at position (l1,c1)
SR600 calculates screen position (l1,c1) of card, sequence number n
SR800 displays the numbered back of a card, sequence number n

PROGRAM LISTING



```
  10 PAPER 7: INK 0: FLASH 0: BO
RDER 7: CLS
  20 FOR i=1 TO 4: READ x$: FOR
j=0 TO 7: READ x: POKE USR x$+j,
x: NEXT j: NEXT i
  30 DATA "H",34,119,127,127,62,
62,28,8,"C",28,28,8,107,127,107,
8,8,"D",8,28,62,127,62,28,8,0,"S
",8,28,62,127,127,127,107,8
  40 LET s$="HCDS": LET v$="A234
56789TJQK"
  50 DIM c(52): DIM t(52): DIM v
(52): DIM s(4)
  60 FOR i=1 TO 4: LET s(i)=CODE
 s$(i)+79: NEXT i
  70 FOR i=1 TO 52: LET c(i)=i:
NEXT i
  80 FOR i=1 TO 52: LET r=INT (5
2*RND)+1: LET z=c(i): LET c(i)=c
(r): LET c(r)=z: NEXT i
  90 FOR i=1 TO 52: LET t(i)=INT
 ((c(i)-1)/13)+1: LET v(i)=c(i)-
13*(t(i)-1): NEXT i
 120 PAPER 4: PRINT TAB 31: FOR
i=1 TO 5: FOR j=1 TO 2: PRINT "
██ ██ ██ ██ ██ ██ ██ ██ ██ ██ ":
 NEXT j: PRINT TAB 31: NEXT i
 130 FOR j=1 TO 2: PRINT " ██ ██
": NEXT j: PRINT TAB 7
 140 PAPER 7: PLOT 0,175: DRAW 0
,-151: DRAW 55,0: DRAW 0,24: DRA
W 192,0: DRAW 0,127: DRAW -247,0
 150 FOR n=1 TO 52: GO SUB 600:
PRINT AT l1,c1; INVERSE 1;n: NEX
T n
 160 LET ch=0: LET pr=0: PRINT A
T 16,8;"CHOICES: 0   PAIRS: 0"
 200 PRINT AT 18,8;"
   ": INPUT "Enter pair: (1)=";p
1
```

```
 210 IF p1<>INT p1 OR p1>52 OR p
1<1 THEN INPUT "Invalid - reente
r (1)=";p1: GO TO 210
 220 IF v(p1)=0 THEN INPUT "Alre
ady paired - reenter (1)=";p1: G
O TO 210
 230 LET n=p1: GO SUB 600: GO SU
B 500
 240 INPUT "             (2)=";p
2
 250 IF p2<>INT p2 OR p2>52 OR p
2<1 THEN INPUT "Invalid - reente
r (2)=";p2: GO TO 250
 260 IF v(p2)=0 THEN INPUT "Alre
ady paired - reenter (2)=";p2: G
O TO 250
 270 IF p1=p2 THEN INPUT "Pair m
ust be separate - reenter (2)";p
2: GO TO 250
 280 LET n=p2: GO SUB 600: GO SU
B 500
 290 LET ch=ch+1: PRINT AT 16,17
;ch
 300 IF v(p1)<>v(p2) THEN GO TO 
340
 310 PRINT AT 18,8; INK 1;"Corre
ct choice!": LET v(p1)=0: LET v(
p2)=0: LET pr=pr+1: PRINT AT 16,
28;pr: BEEP .5,25: BEEP .5,15
 320 IF pr=26 THEN PRINT AT 20,0
;"You have finished after ";ch;"
 turns": BEEP .5,20: BEEP .5,25:
 BEEP .5,20: GO TO 360
 330 GO TO 350
 340 PRINT AT 18,8;"Do not match
  ": BEEP 1,-15: PAUSE 150: GO S
UB 800: LET n=p1: GO SUB 600: GO
 SUB 800
 350 INPUT "Press ENTER for next
 go"; LINE z$: GO TO 200
 360 PAUSE 150: INPUT "Press 1 t
o repeat or 0 to stop"; LINE z$
 370 IF z$="1" THEN RUN
 380 STOP
 500 PAPER 7: LET vn=v(n): LET s
n=t(n): INK 2-(sn=2*INT (sn/2))*
2
 510 PRINT AT l1,c1;v$(vn);" ";A
T l1+1,c1;" ";CHR$ s(sn)
 520 INK 0
 530 PLOT c1*8-1,176-8*l1: DRAW 
17,0: DRAW 0,-17: DRAW -17,0: DR
AW 0,17
 540 BEEP 1,5: RETURN
 600 LET x10=INT (n/10): LET x1=
n-10*x10
 605 IF x1=0 THEN LET x10=x10-1:
 LET x1=10
 610 LET l1=1+3*x10
```

```
620 LET c1=3*x1-2
630 RETURN
800 PRINT AT l1,c1; INVERSE 1;n
: IF n<10 THEN PRINT "█"
810 PRINT AT l1+1,c1;"██"
820 RETURN
```

NOTES

1. It is important that the player is not allowed to pick a card that has already been paired. These cards that have already been paired and turned face up must be distinguished in some way: this is done by setting the appropriate value numbers to 0, and when a card is chosen by the player it is only allowed if the value number is non-zero (see lines 310 and 220).

## 8.3 BLACKJACK

### DESCRIPTION

Blackjack has several different variations but the basic aim is to arrive at a hand of cards the values of which total twenty-one or as close below twenty-one as possible. The player starts with two cards, and thereafter has a choice of whether to accept another card (to TWIST) or not (to STICK). If the player's hand exceeds twenty-one, the result is a BUST.

There are five sets of hands in each game and there is a regulated bet of $10. The player starts, and if the hand does not exceed twenty-one, the "house" (i.e. the TS2068) plays next. The house's strategy is not complex and can either be worked out from playing or from the program itself. If the player wins, the prize is $50, while if it is a draw, the player regains the bet (i.e. $10). Cards are valued between 1 and 13 with the ace always being low.

### METHOD

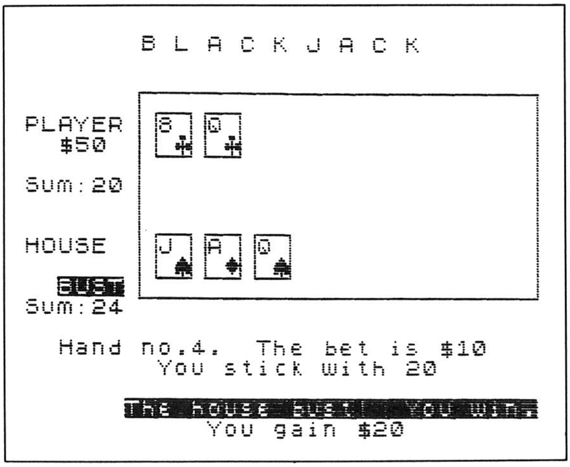
1. Initialize screen and variables
   - a) Screen colors (10)
   - b) Set up graphics for suit symbols (20-30)
   - c) Set up suit initials (s$) and value symbols (v$) (40)
   - d) Set up array s containing graphic symbol codes (60)
   - e) Set up cards 1-52 in array c (70)
   - f) Shuffle cards (80)
   - g) Set up card values (array v) and suits (array t) (90)

```
    h) Set up screen layout                          (100-120, 140)
    i) Set money m to 50 and display                         (130)
    j) Set bet (b) to 10                                     (130)
    k) Set card at top of pack (cd) to 1                     (130)

2.  For go number g from 1 to 5                              (200 )
    a) Clear message lines and card table on screen          (200 )
    b) Display go number and bet                             (210)
    c) Decrease money (m) by bet (b)              (220 and SR650)
    d) Players go:                                           (240)
       (i)  Display two cards for player                   (SR800)
       (ii) Calculate sum (pt) and display
       (iii) If sum > 21 then                                (250)
              A) Message 'BUST — house wins'
              B) Go to (9)
       (iv) If sum = 21 then                                 (260)
              A) Message 'EQUALS 21'
              B) Go to (e)
       (v)  Enter value z$                                   (270)
       (vi) If z$ not ENTER then (TWIST):
              A) Display another card             (280 and SR850)
              B) Add card value to sum (pt) and display
              C) Go back to d(iii)
       (vii) (STICK)
              A) Message 'STICK' and display sum pt          (300)
    e) Wait                                                  (310)
    f) House go:                                             (320)
       (i)  Display two cards for house                    (SR800)
       (ii) Calculate sum (ht) and display
       (iii) If sum > 21 then:                               (330)
              A) Message 'BUST — you win'
              B) Go to f (vii)
       (vi) If house sum < = player's sum and                (340)
            house sum < 19 then (TWIST):
              A) Display another card                      (SR850)
              B) Add card value to sum (ht) and display
              C) Go back to f (iii)
       (v)  If house sum (ht) > player's sum (pt) then       (350)
              A) Message 'HOUSE WINS'
              B) Go to (g)
       (vi) If house sum (ht) = player's sum (pt) then       (360)
              A) Message 'DRAW'
              B) Add bet (b) to money (m)                  (SR650)
              C) Go to (g)
```

(vii) (PLAYER WINS)
A) Message 'WIN' and display amount (370)
B) Add twice bet (2*b) to money (m)
g) Press ENTER (380)
h) Repeat to (2) (390)
3. End
a) Message showing money left (390)
b) Rerun or stop (400-410)

SCREEN LAYOUT



```
         B L A C K J A C K

PLAYER
 $50

Sum:20

HOUSE

  BUST
Sum:24

   Hand no.4.  The bet is $10
        You stick with 20

       The house bust! You win!
             You gain $20
```

VARIABLES

i,j = general purpose counters
x$ = graphic character
x = binary pattern for graphic character
s$ = suit initials, H,C,D,S
v$ = values of cards A,2,3,...,9,T,J,Q,K
array c = holds the numbers 1-52 representing individual cards
array t = holds the suit numbers of the cards in array c
array v = holds the value numbers of the cards in array c
array s = holds the codes of the graphic characters for the suits
r = a random number (1-52) used in shuffling the cards
z,z$ = dummy variables
m = money held (initially 50)
b = standard amount of bet

cd = sequence number of card currently at top of pack i.e. next available card
g = go number
incm = amount by which money (m) is to be changed
playersgo = true if it is the player's go, false if the houses's go
pt = player's sum of card values
ht = house's sum of card values
n = sequence number of card displayed by SR500
sn = suit number of card n
vn = value number of card n
l1,c1 = screen position of top left hand corner of card n
n1 = new value of money (m)
cd1 = sequence number of first card in player's/house's hand
total = sum of hand
inct = amount by which sum is to be incremented.

## ROUTINES

SR500 = displays card number n with top left hand corner at screen line l1, column c1
SR650 = increments money (m) by incm, as shown on screen
SR800 = deals two cards from the current top of the pack to either the player or the house. Calculates sum of cards.
SR850 = (TWIST) deals one more card from the current top of the pack to either the player or the house.
SR900 = clears screen message areas and 'card table' area.

## PROGRAM LISTING

```
  10 PAPER 7: INK 0: FLASH 0: BO
RDER 7: CLS
  20 FOR i=1 TO 4: READ x$: FOR
j=0 TO 7: READ x: POKE USR x$+j,
x: NEXT j: NEXT i
  30 DATA "H",34,119,127,127,62,
62,28,8,"C",28,28,8,107,127,107,
8,8,"D",8,28,62,127,62,28,8,0,"S
",8,28,62,127,127,127,107,8
  40 LET s$="HCDS": LET v$="A234
56789TJQK"
  50 DIM c(52): DIM t(52): DIM v
(52): DIM s(4)
  60 FOR i=1 TO 4: LET s(i)=CODE
 s$(i)+79: NEXT i
  70 FOR i=1 TO 52: LET c(i)=i:
NEXT i
  80 FOR i=1 TO 52: LET r=INT (5
2*RND)+1: LET z=c(i): LET c(i)=c
(r): LET c(r)=z: NEXT i
```

```
  90 FOR i=1 TO 52: LET t(i)=INT
 ((c(i)-1)/13)+1: LET v(i)=c(i)-
13*(t(i)-1): NEXT i
 100 GO SUB 900
 110 PLOT 55,152: DRAW 193,0: DR
AW 0,-81: DRAW -193,0: DRAW 0,81
 120 PRINT INK 1;AT 4,0;"PLAYER"
;AT 10,0;"HOUSE"
 130 LET m=50: PRINT AT 5,2;"$";
m: LET b=10: LET cd=1
 140 PRINT AT 0,7;"B L A C K J A
 C K"
 200 FOR g=1 TO 5: GO SUB 900: P
AUSE 100
 210 PRINT AT 15,2;"Hand no.";g;
".  The bet is $";b
 220 PAUSE 150
 230 LET incm=-b: GO SUB 650
 240 LET playersgo=1: GO SUB 800
: LET pt=total
 250 IF pt>21 THEN PRINT AT 6,2;
 INVERSE 1; INK 2;"BUST": PRINT
AT 18,6;"You bust!  The house wi
ns.": GO TO 380
 260 IF pt=21 THEN PRINT AT 16,6
;"You have exactly 21!": GO TO 3
10
 270 INPUT "Press ENTER to stick
, any other key to twist"; LINE
z$: IF z$="" THEN GO TO 300
 280 GO SUB 850: LET pt=pt+inct:
 PRINT AT 7,4;pt: GO TO 250
 300 PRINT AT 16,8;"You stick wi
th ";pt
 310 PAUSE 200
 320 LET playersgo=0: GO SUB 800
: LET ht=total
 330 IF ht>21 THEN PRINT AT 12,2
; INVERSE 1; INK 2;"BUST";AT 18,
6;"The house bust!  You win.": G
O TO 370
 340 IF ht<=pt AND ht<19 THEN GO
 SUB 850: LET ht=ht+inct: PRINT
AT 13,4;ht: GO TO 330
 350 IF ht>pt THEN PRINT AT 18,8
;"The house wins": GO TO 380
 360 IF ht=pt THEN PRINT AT 18,9
;"It is a draw": LET incm=b: GO
SUB 650: GO TO 380
 370 PRINT AT 19,11;"You gain $"
;2*b: LET incm=2*b: GO SUB 650
 380 INPUT "Press ENTER to conti
nue"; LINE z$
 390 NEXT g: PRINT AT 20,3;"That
's it.  You have $";m
 400 PAUSE 150: INPUT "Press ENT
ER to play again, or 0 to stop";
 LINE z$: IF z$="" THEN RUN
 410 STOP
 500 PAPER 7: LET vn=v(n): LET s
n=t(n): INK 2-(sn=2*INT (sn/2))*
2
 510 PRINT AT l1,c1;v$(vn);" ";A
T l1+1,c1;" ";CHR$ s(sn)
 520 INK 0
 530 PLOT c1*8-1,176-8*l1: DRAW
17,0: DRAW 0,-17: DRAW -17,0: DR
AW 0,17
 540 BEEP 1,5: RETURN
 650 LET m1=m+incm
 660 FOR i=m TO m1 STEP 1-2*(m1<
m)
 670 PRINT AT 5,3;i;" ": BEEP .0
1,15
 680 NEXT i
 690 LET m=m1: RETURN
 800 LET l1=10
 810 IF playersgo THEN LET l1=4
 820 LET c1=8: LET n=cd: LET cd1
=cd: LET cd=cd+1: GO SUB 500: LE
T c1=11: LET n=cd: LET cd=cd+1:
GO SUB 500
 830 LET total=v(cd-1)+v(cd-2):
PRINT AT l1+3,0; INK 1;"Sum:";to
tal
 840 RETURN
 850 PRINT AT 17,13; INK 3; FLAS
H 1;"TWIST": PAUSE 150: PRINT AT
 17,13;"     "
 860 LET c1=3*(cd-cd1+1)+5: LET
n=cd: GO SUB 500
 870 LET inct=v(cd): LET cd=cd+1
 880 RETURN
 900 FOR i=15 TO 19: PRINT AT i,
0;"
   ": NEXT i
 910 PRINT AT 6,2;"      ";AT 7,0;
"       ";AT 12,2;"      ";AT 13,0;
"       "

 920 FOR i=3 TO 12: PRINT PAPER
4;AT i,7;"
  ": NEXT i
 930 RETURN
```

NOTES

1. In this game, as in any where the dealer does not deal out all the cards at first, the position of the current top of the pack must be carefully kept. Variable cd is used for this purpose in the program above, and it is incremented whenever a card is dealt. Variable cd1 is also needed when dealing cards to a particular player to show how many cards he has received so far, and thus indicates where the next card is to be displayed on the screen - in the third, fourth, fifth or whatever position.

# CHAPTER NINE

# GRID GAMES

## 9.1 TREASURE TROVE

### DESCRIPTION

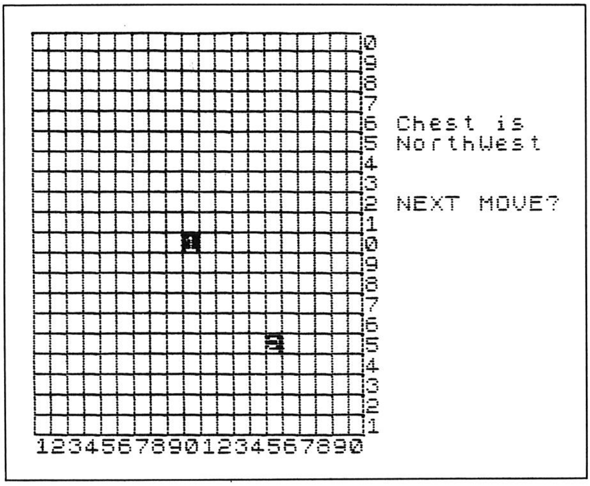
Many computer games are based on some form of grid or matrix. In this chapter we consider increasingly sophisticated games all based upon the grid principle. Treasure is the simplest version of a grid game, and in fact the grid itself is only conceptual—it does not need to be stored in the computer.

A treasure chest is hidden in a network of caverns and your task is to find it. You specify where you want to be in the network by entering a horizontal and vertical position. You have five attempts, and after each one the computer tells you in which direction to move next. You may choose the size of the cavern network at the beginning of the game.

### METHOD

```
1.  Initialize screen and variables
    a) Set screen colors                              (10)
    b) Set number of turns (ng)                       (20)
    c) Enter size of grid (gs)                        (30)
    d) If size (gs) invalid then                      (40)
       (i)  Message and reenter
       (ii) Go to (d)
    e) Draw grid                                   (50-90)
    f) Choose treasure position (tx, ty)             (100)
2.  For go number g from 1 to ng                     (110)
    a) Enter player's move (px, py)              (120-140)
    b) If px, py invalid then                      (SR500)
       (i)  Message and reenter
       (ii) Go to (b)
    c) Display player's position (px, py)            (150)
    d) If player's position = treasure position then     (160)
        (i) Message of congratulation
        (ii) Go to 3(b)
    e) Message regarding direction to proceed             (170-210)
    f) Repeat to (2)                                      (220)
3.  End
    a) Message of commiseration                           (230)
    b) Highlight treasure position                        (240)
    c) Rerun or stop                                      (250-260)
```

SCREEN LAYOUT



```
Chest is
NorthWest

NEXT MOVE?
```

VARIABLES

ng = number of turns allowed
gs = dimension of grid
l = line counter
c = column counter
ld = length of lines DRAWn around grid
y = y-co-ordinate counter
n = parameter for FUNCTION d
tx, ty = position of treasure
g = go counter
pos = parameter for SUBROUTINE 500
px, py = player's position
z$ = dummy variable, = 1 to rerun

## ROUTINES

SR500 = checks that an input value (pos) is between 1 and gs
FN d = removes first digit from a 2-digit decimal number (n)

## PROGRAM LISTING

```
   10 BORDER 7: PAPER 7: INK 0: F
LASH 0: CLS
   20 LET ng=5
   30 INPUT "Choose size of netwo
rk of caves (10-20):";gs
   40 IF gs<>INT gs OR gs<10 OR g
s>20 THEN INPUT (gs;" is incorre
ct. Reenter (10-20):");gs: GO TO
 40
   50 FOR l=0 TO gs-1: FOR c=0 TO
 gs-1: PRINT PAPER 5;AT l,c;" ";
: NEXT c: PRINT : NEXT l
   60 LET ld=gs*8-1: FOR y=175-ld
 TO 175 STEP 8: PLOT 0,y: DRAW l
d,0: NEXT y: FOR x=0 TO ld STEP
8: PLOT x,175-ld: DRAW 0,ld: NEX
T x: PLOT 0,175: DRAW ld,0: DRAW
 0,-ld
   70 DEF FN d(n)=n-10*INT (n/10)
   80 FOR l=0 TO gs-1: PRINT AT l
,gs;FN d(gs-l): NEXT l
   90 FOR c=0 TO gs-1: PRINT AT g
s,c;FN d(c+1): NEXT c
  100 LET tx=INT (RND*gs)+1: LET
ty=INT (RND*gs)+1
  110 FOR g=1 TO ng
  120 PRINT AT 8,22; FLASH 1;"NEX
T MOVE?"
  130 INPUT "Horizontal: ";px: LE
T pos=px: GO SUB 500: LET px=pos
  140 INPUT "Vertical: ";py: LET
pos=py: GO SUB 500: LET py=pos
  150 PRINT AT 8,22; FLASH 0;"
       ";AT gs-py,px-1; INVERSE
1;g
  160 IF px=tx AND py=ty THEN PRI
NT AT 4,22;"WELL DONE!";AT 5,22;
"You got it": BEEP .5,15: BEEP .
5,25: BEEP .5,15: GO TO 240
  170 PRINT AT 4,22;"Chest is";AT
 5,22;"          ";AT 5,22;
  180 IF py<ty THEN PRINT "North"
;
  190 IF py>ty THEN PRINT "South"
;
  200 IF px<tx THEN PRINT "East"
  210 IF px>tx THEN PRINT "West"
  220 NEXT g
 230 PRINT AT 8,22;"The end.";AT
 9,22;"It was at";AT 10,22;"(";t
x;",";ty;")"
 240 PRINT AT gs-ty,tx-1; INK 2;
 FLASH 1;"*"
 250 INPUT "Enter 1 to repeat, 0
 to stop"; LINE z$: IF z$="1" TH
EN RUN
 260 STOP
 500 IF pos<>INT pos OR pos>gs O
R pos<1 THEN INPUT "Out of range
.  Reenter ";pos: GO TO 500
 510 RETURN
```

Although a complete grid is shown on the screen, the theory of the game is simply the choice of two random numbers. In the more sophisticated games that follow we see how this simple concept may be improved upon.

## 9.2 GRAB THE GOBLIN

### DESCRIPTION

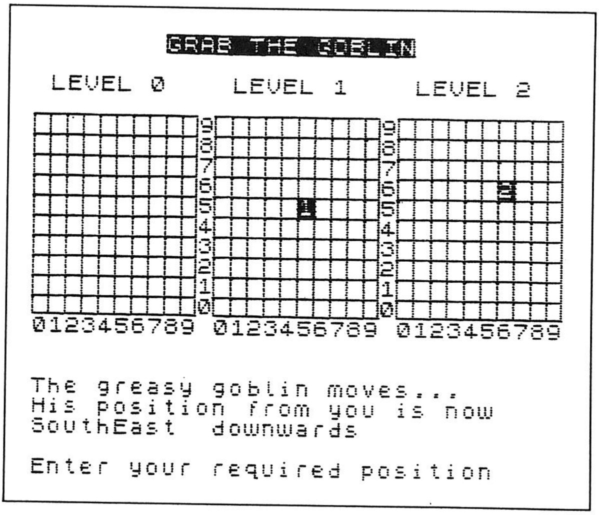
Many grid games involve the tracking down of a creature. In Grab the Goblin an irksome goblin is hiding in a three-dimensional network of tunnels. The goblin can move north, south, east or west, or combinations of these, and up or down, but only one position at a time. In this game the size of the tunnels is fixed: there are three levels of tunnels, ten by ten square. The player again has five attempts to find the goblin, whose movement follows no set pattern. The goblin does occasionally however display extreme aggression when cornered. Note that the grid axes are labelled 0-9 this time, rather than from 1.

### METHOD

1. Initialize screen and variables
   a) Set screen colors
   b) Set number of goes (ng)
   c) Draw format with three grid levels (30-90 and SRs700 and 710)
   d) Set initial goblin position (gx,gy,gl) (100)
2. For go number g from 1 to ng (110)
   a) Enter player's move (px,py,pl) (120-160)
   b) If px,py,pl invalid then (SR500 and 160)
      (i) Message and reenter

    - (ii) Go to (b)
  - c) Display player's position (px,py,pl) (170)
  - d) If player's position = goblin's position then (180)
    - (i) Message of congratulation
    - (ii) Go to 3(d)
  - e) Change goblin's position (gx,gy,gl) at random (190-230 and SR600)
  - f) If goblin's position = player's position then (240)
    - (i) Message concerning attack
    - (ii) Go to 3(c)
  - g) Calculate and display direction of goblin from player
  - h) If goblin close to player then display warning (320)
  - i) Repeat to (2)
- 3. End
  - a) End message
  - b) Reveal goblin's position
  - c) End sound (350)
  - d) Display goblin (355)
  - e) Rerun or stop (360)

SCREEN LAYOUT



```
The greasy goblin moves...
His position from you is now
SouthEast  downwards

Enter your required position
```

## VARIABLES

ng = number of goes allowed
l = line counter
x = x-co-ordinate counter
y = y-co-ordinate counter
n = counter for labelling y-axis
gx,gy,gl = position of goblin
px,py,pl = position of player
pos = parameter for subroutine 500
gc = parameter for subroutine 600
g = go counter
i = general purpose counter
z$ = dummy variable, = 1 to rerun

## ROUTINES

SR500 = checks that an input value (pos) is between 0 and 9
SR550 = clears message lines
SR600 = increments a goblin co-ordinate by −1, 0 or +1
SR700 and SR710 = used in drawing grids
FN g = true if goblin and player at same positions, otherwise false

## PROGRAM LISTING

```
  10 BORDER 7: PAPER 7: INK 0: F
LASH 0: CLS
  20 LET ng=5
  30 PRINT AT 0,8; INK 2; INVERS
E 1;"GRAB THE GOBLIN"
  40 PRINT AT 2,1;"LEVEL 0    LE
VEL 1    LEVEL 2"
  50 FOR l=4 TO 13: PRINT PAPER
4;AT l,0;"          ";AT l,11; P
APER 5;"          ";AT l,22; PAP
ER 6;"          ": NEXT l
  60 FOR x=0 TO 255 STEP 8: GO S
UB 700: NEXT x: FOR x=79 TO 255
STEP 88: GO SUB 700: NEXT x
  70 FOR y=143 TO 64 STEP -8: GO
 SUB 710: NEXT y: LET y=64: GO S
UB 710
  80 FOR n=9 TO 0 STEP -1: PRINT
 AT 13-n,10;n;AT 13-n,21;n: NEXT
 n
  90 PRINT AT 14,0;"0123456789 0
123456789 0123456789"
```

```
 100 LET gx=INT (RND*10): LET gy
=INT (RND*10): LET gl=INT (RND*3
): DEF FN g()=(px=gx) AND (py=gy
) AND (pl=gl)
 110 FOR g=1 TO ng
 120 PRINT AT 21,0; FLASH 1;"Ent
er your required position"
 130 INPUT "Horizontal: ";px: LE
T pos=px: GO SUB 500: LET px=pos
 140 INPUT "Vertical: ";py: LET
pos=py: GO SUB 500: LET py=pos
 150 INPUT "Level: ";pl
 160 IF pl<>INT pl OR pl>2 OR pl
<0 THEN INPUT "Out of range.  Re
enter ";pl: GO TO 160
 170 GO SUB 550: PRINT AT 13-py,
px+11*pl; INVERSE 1;g
 180 IF FN g() THEN GO SUB 550:
PRINT AT 17,0; INK 2;"Congratula
tions.  You caught him": BEEP .5
,15: BEEP .5,25: BEEP .5,15: GO
TO 355
 190 LET gc=gx: GO SUB 600: LET
gx=gc
 200 LET gc=gy: GO SUB 600: LET
gy=gc
 210 LET gc=gl: GO SUB 600: LET
gl=gc
 220 IF gl>2 THEN LET gl=2
 230 GO SUB 550: PRINT AT 17,0;"
The greasy goblin moves...": PAU
SE 150: PRINT AT 18,0;

 240 IF FN g() THEN PRINT "Oh no
!  He has moved to your"'"positi
on and drawn a knife!"'"I'm sorr
y, you're done for.": GO TO 350
 250 PRINT "His position from yo
u is now";AT 19,0;"
                 ";AT 19,0;

 260 IF py<gy THEN PRINT "North"
;
 270 IF py>gy THEN PRINT "South"
;
 280 IF px<gx THEN PRINT "East  "
;
 290 IF px>gx THEN PRINT "West  "
;
 300 IF pl<gl THEN PRINT " upwar
ds"
 310 IF pl>gl THEN PRINT " downw
ards"
 320 IF ABS (px-gx)+ABS (py-gy)+
ABS (pl-gl)<5 THEN PRINT "Be car
eful, he is close by!"
 330 NEXT g
 340 PRINT INVERSE 1;AT 21,0;"Th
e end. He is at level ";gl;" (";
gx;",";gy;")"
```

```
 350 FOR i=0 TO -20 STEP -1: BEE
P .1,i: NEXT i: BEEP 3,-20
 355 PRINT AT 13-gy,gx+gl*11; FL
ASH 1; INK 2;"*"
 360 INPUT "Enter 1 to repeat, 0
 to stop"; LINE z$: IF z$="1" TH
EN RUN
 370 STOP
 500 IF pos<>INT pos OR pos>9 OR
 pos<0 THEN INPUT "Out of range.
  Reenter ";pos: GO TO 500
 510 RETURN
 550 FOR l=17 TO 21: PRINT AT l,
0;"
   ": NEXT l: RETURN
 600 LET gc=gc-1+INT (RND*3): IF
 gc<0 THEN LET gc=0
 610 IF gc>9 THEN LET gc=9
 620 RETURN
 700 PLOT x,143: DRAW 0,-79: RET
URN
 710 PLOT 0,y: DRAW 79,0: PLOT 8
8,y: DRAW 79,0: PLOT 176,y: DRAW
 79,0: RETURN
```

NOTES

1. Once again an array is not needed to store the grids, but here more variables are needed to keep track of the goblin and the player, since we are now working in three dimensions.

2. As a result of the randomizing routine, the goblin does tend to stay near a grid edge or the top or bottom levels more often than not.

## 9.3 PURSUIT

DESCRIPTION

If you derive pleasure from being chased around by malevolent monsters, you should enjoy Pursuit. It is particularly a game for masochists, because it is also extremely difficult to win.

For unexplained reasons you find yourself in the lair of a group of Gulpers, strange creatures whose sole object in life is to find and devour anything edible, and both humans and other Gulpers come into that category. The lair is totally surrounded by an antimatter barrier, lethal to all forms of life. Finally there are also a few Spongs scattered around the lair. A Spong shows little signs of activity until you step on it, at which point it dematerializes you. Spongs show no discrimination, absorbing both Gulpers and humans alike. Your objective is to avoid these many and various hazards for ten turns. On each turn, you move one step in any direction and the Gulpers try and catch you. Wise players will attempt to lure Gulpers towards a Spong so that there are less Gulpers to avoid. A move is specified as an abbreviated direction i.e. N,S,E,W,NE,SE,NW or SW. Use * to give up. To change the level of difficulty, alter the number of Spongs, Gulpers, or turns in line 10.

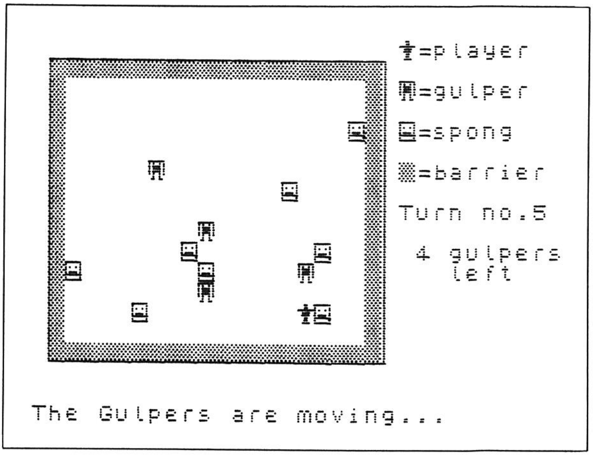
METHOD

```
1. Initialize screen and variables
   a) Set screen colors                                          (10)
   b) Set number of turns (nt), number of Spongs (ns),
      number of Gulpers (ng) and number of Gulpers
      left (ngl)                                                 (20)
   c) Set up user graphic symbols                             (50-70)
   d) Display symbol key and headings                            (80)
   e) Draw lair                                              (90-150)
   f) Draw ns Spongs randomly                      (160 and SR500)
   g) Set position of Gulpers in arrays x and y and
      display                                      (170 and SR500)
   h) Set position of player (px,py) and
      display                                  (180-190 and ST500)
2. For turn number t from 1 to nt                              (200)
   a) Display turn number                                      (200)
   b) Enter player's move m$                                   (205)
   c) If m$ is "*" then                                        (207)
      (i) Quit message and sound                        (and SR900)
      (ii) Go to 3(b)
   d) If direction m$ invalid then                 (210 and SR600)
      (i) Message and reenter
      (ii) Go to (d)
   e) Calculate players new position (px,py)       (220 and SR600)
   f) If object at (px,py) then                                (230)
      (i) Display message that player hit gulper/spong/
          barrier                                          (240-270)
      (ii) Sound                                           (SR900)
      (iii) Go to 3(b)                                       (270)
   g) Display player's new position (px,py)                    (280)
   h) Clear message lines                                    (SR800)
   i) For every Gulper:                                        (300)
      (i) If eliminated (ie. line position = 0) then go to (vii) (305)
      (ii) Recalculate Gulpers position to move closer to
           player                                              (320)
          (iii) If new position = player's position then          (340)
                A) Death message and sound                  (and SR900)
                B) Go to 3(b)
          (iv) If new position = object then                       (345)
                A) Message that Gulper has been Sponged, or
                   Barred and sound              (350-365 and SR850)
                B) Decrease Gulpers left (ngl) by 1
                C) Set Gulper's line position x to 0
                D) Go to (vi)
          (v)  Display Gulper and sound                        (370-375)
          (vi) Display number of Gulpers left (ngl)                (380)
          (vii) Repeat to (i)                                      (390)
        j) If no Gulpers left (ngl=0) then                         (400)
          (i)  Message
          (ii) Go to 3(b)
        k) Repeat to (2)                                           (410)
    3.  End
        a) Survived message                                        (420 )
        b) Rerun or stop                                       (430-450)
```

VARIABLES

nt = number of turns
ns = number of Spongs
ng = initial number of Gulpers
ngl = number of Gulpers left
arrays x,y = two arrays holding the line and column positions of each Gulper. If a Gulper is destroyed its x-value is set to 0
i,j,k = general purpose counters or function parameters
x$ = used to hold character for setting up user graphics
x = used to hold binary value for setting up user graphics
l,c = line and column positions found to be free (by SR500)
px,py = player's line and column position
t = turn counter
m$ = direction to move (= N/S/E/W/NE/SE/NW/SW/*)
ok = 1 if m$ is invalid, 0 if invalid
ix = line increment by which the player's position is to be changed to move
iy = column increment by which the player's position is to be changed to move
w$ = word showing object player walked into
lm = length of direction entry m$
y$ = parameter used by SR630 - a single direction letter

## SCREEN LAYOUT



```
=player
=gulper
=spong
=barrier
Turn no.5
 4 gulpers
   left

The Gulpers are moving...
```

## ROUTINES

SR500 = finds a position (l,c) in the lair not occupied by a Spong, Gulper or barrier
SR600 = test whether m$ is a valid direction. Sets ok to 0 if invalid or 1 if valid. Calls SR700 and SR750
SR700 = calculates increment for px, based upon direction m$
SR750 = calculates increment for px, based upon direction m$
SR800 = clears message lines
SR850 = funeral sound for dead Gulper
SR900 = funeral sound for dead player
FN l = gives a random line position within the lair (2-14)
FN c = gives a random column position within the lair (2-19)
FN i = using FN b, extracts the ink colour of a position (j,k) in the lair
FN b = as above

## PROGRAM LISTING

```
  10 BRIGHT 0: FLASH 0: PAPER 7:
 BORDER 7: INK 0: CLS
  20 LET nt=10: LET ns=8: LET ng
=6: LET ngl=ng: DIM x(ng): DIM y
(ng)
  30 DEF FN l()=INT (RND*13)+2
  40 DEF FN c()=INT (RND*18)+2
  43 DEF FN b(n)=(n-64)-8*INT ((
n-64)/8)
  45 DEF FN i(j,k)=FN b(ATTR (j,
k))
  50 FOR i=1 TO 4: READ x$
  60 FOR j=0 TO 7: READ x: POKE
USR x$+j,x: NEXT j: NEXT i
  70 DATA "B",170,85,170,85,170,
85,170,85,"G",255,165,189,189,18
9,165,36,102,"P",24,24,255,60,60
,24,24,60,"S",255,129,165,129,18
9,255,129,255
  80 PRINT AT 0,22;"P=player";AT
 2,22; INK 2;"G=gulper";AT 4,22;
 INK 1;"S=spong";AT 6,22;"B=barr
ier"; INK 0;AT 8,22;"Turn no.1";
AT 10,23;ng;" gulpers";AT 11,25;
"left"
  90 INK 1: PAPER 6
 100 FOR l=2 TO 14: PRINT AT l,2
;"                  ": NEXT l
 110 BRIGHT 1
 120 PRINT AT 1,1;"BBBBBBBBBBBBB
BBBBBBB";AT 15,1;"BBBBBBBBBBBBBB
BBBBBB"
 130 FOR l=2 TO 14: PRINT AT l,1
;"B";AT l,20;"B": NEXT l
 140 INK 0: PAPER 8
 150 PLOT 8,167: DRAW 161,0: DRA
W 0,-119: DRAW -161,0: DRAW 0,11
9
 160 FOR i=1 TO ns: GO SUB 500:
PRINT AT l,c; INK 1; PAPER 7;"S"
: NEXT i
 170 FOR i=1 TO ng: GO SUB 500:
LET x(i)=l: LET y(i)=c: PRINT AT
 l,c; INK 2;"G": NEXT i
 180 BRIGHT 0
 190 GO SUB 500: LET px=l: LET p
y=c: PRINT AT px,py;"P"
 200 FOR t=1 TO nt: PRINT AT 8,3
0;t

 205 INPUT "Enter direction to m
ove:";m$
 207 IF m$="*" THEN PRINT FLASH
1;AT 20,0;"You gave up": GO SUB
900: GO TO 430
 210 GO SUB 600: IF NOT ok THEN
INPUT "Reenter (eg. SW or N):";m
$: GO TO 210
 220 PRINT AT px,py; PAPER 6;" "
: LET px=px+ix: LET py=py+iy:
 230 IF ATTR (px,py)<64 THEN GO
TO 280
 240 IF FN i(px,py)=2 THEN LET w
$="Gulper": GO TO 270
 250 IF FN i(px,py)=1 THEN LET w
$="Spong": GO TO 270
 260 LET w$="Barrier"
 270 PRINT AT px,py; FLASH 1;"*"
; INK 3;AT 18,0;"You walked into
 a ";w$;"!": GO SUB 900: GO TO 4
30
 280 PRINT AT px,py;"P": GO SUB
800: BEEP .2,20: BEEP .2,10
 290 PRINT AT 18,0;"The Gulpers
are moving..."
 300 FOR i=1 TO ng
 305 IF x(i)=0 THEN GO TO 390
 310 PRINT AT x(i),y(i); PAPER 6
;" "
 320 LET x(i)=x(i)+SGN (px-x(i))
: LET y(i)=y(i)+SGN (py-y(i))
 340 IF px=x(i) AND py=y(i) THEN
 PRINT AT px,py; FLASH 1;"G"; IN
K 3;AT 20,0;"Gulper ";i;" has go
t you!": GO SUB 900: GO TO 430
 345 IF ATTR (x(i),y(i))<64 THEN
 GO TO 370
 350 IF FN i(x(i),y(i))=1 THEN P
RINT AT x(i),y(i); FLASH 1; INK
1; PAPER 7;"S";AT 19,0; FLASH 0;
"Gulper ";i;" absorbed    ": GO
SUB 850: PRINT AT x(i),y(i); BRI
GHT 1; INK 1; PAPER 7;"S": LET n
gl=ngl-1: LET x(i)=0: GO TO 380
 360 IF FN i(x(i),y(i))=0 THEN P
RINT AT x(i),y(i); FLASH 1;"*";
FLASH 0;AT 19,0;"Gulper ";i;" me
lted      ": GO SUB 850: PRINT A
T x(i),y(i); BRIGHT 1; PAPER 6;"
B": LET ngl=ngl-1: LET x(i)=0: G
O TO 380
 365 IF FN i(x(i),y(i))=2 THEN P
RINT AT x(i),y(i); FLASH 1;"G";A
T 19,0; FLASH 0;"Gulper ";i;" ca
nnibalized": GO SUB 850: PRINT A
T x(i),y(i); BRIGHT 1; INK 2; PA
PER 6;"G": LET ngl=ngl-1: LET x(
i)=0: GO TO 380
 370 PRINT AT x(i),y(i); BRIGHT
1; INK 2;"G"

 375 BEEP .1,15: PAUSE 25: BEEP
.1,5: BEEP .1,-5
 380 PRINT AT 10,23;ngl
 390 NEXT i
 400 IF ngl=0 THEN PRINT FLASH 1
;AT 20,0;"No Gulpers left - you
win!": GO TO 430
 410 NEXT t
 420 PRINT FLASH 1;AT 20,0;"Well
 done - you have survived!"
 430 PRINT AT 10,23; FLASH 1;ngl
;" gulpers";AT 11,25;"left": PAU
SE 200: INPUT "Enter 1 to repeat
, 0 to stop"; LINE z$: IF z$="1"
 THEN RUN
 450 STOP
 500 LET l=FN l(): LET c=FN c()
 510 IF ATTR (l,c)>64 THEN GO TO
 500
 520 RETURN
 600 LET ok=0: LET lm=LEN m$
 610 IF lm<1 OR lm>2 THEN RETURN

 620 IF lm=1 THEN LET y$=m$(1):
GO SUB 700: GO SUB 750: IF ix=0
AND iy=0 THEN RETURN
 630 IF lm=2 THEN LET y$=m$(1):
GO SUB 700: LET y$=m$(2): GO SUB
 750: IF ix=0 OR iy=0 THEN RETUR
N
 640 LET ok=1: RETURN
 700 LET ix=(y$="S" OR y$="s")-(
y$="N" OR y$="n"): RETURN
 750 LET iy=(y$="E" OR y$="e")-(
y$="W" OR y$="w"): RETURN
 800 FOR l=18 TO 20: PRINT AT l,
0;"
  ": NEXT l
 810 RETURN
 850 FOR j=1 TO 3: BEEP 1,8*j: B
EEP .01,20: NEXT j
 860 RETURN
 900 FOR j=1 TO 4: BEEP .1,20: B
EEP 1,-20: NEXT j: BEEP 5,-20
 910 RETURN
```

**(Note: The letters P, G, S and B (lines 80, 120, 130, 160, 170, 190, 280, and 340-370) must be entered in graphics mode (cursor G).)**

## NOTES

1. The "test for a hit" routines for both Gulpers and player again use the ATTR function to test for BRIGHT characters. Gulpers, Spongs and the barriers are the only BRIGHT items on the grid, so this test works perfectly. However, distinguishing a Spong from a Gulper and so on is more difficult. The ink color value of an attribute byte is held in bits 2 to 0 and this value is extracted by function FN b. Providing Spongs, Gulpers and barriers have their own distinguishing color (blue, red and black respectively) then the object hit may be thus clearly identified.

2. The routines which check the user's entry for a valid direction also calculate the effect in the player's line and column positions. These routines SR600, SR700 and SR750 merit further attention. SR700 sets the line increment (ix) to 1 for N, −1 for S and 0 otherwise. Similarly SR750 sets the column increment (iy) to 1 for E,

—1 for W and 0 otherwise. SR600 uses both these routines in testing the user's input. If the input is one character then both routines are called, and if ix and iy are both 0 then the directions cannot have been N, S, E or W so it is invalid. If the input is two characters then SR700 operates on the first character (must be N or S) and SR750 on the second (E or W). If either ix or iy are still 0 then something is wrong so the entry is flagged as invalid.

# INDEX

ATTR 66,70,118
*Awful Anagrams* 29

BEEP table 20
*Blackjack* 48
**BOARD GAMES** 47

**CARD GAMES** 87
Cassette interface 5
Comments 14
*Concentration* 93
Control Characters 42
*Copycat* 47
*Crack the Combination* 17

Design 9
*Dead Ducks* 66
**DESIGNING TS2068 GAMES** 7
**DICE GAMES** 71

Functions 58

Games—general 7
Good Game 8
*Grab the Goblin* 108
**GRID GAMES** 105

*Headquarters* 61
*Higher-Lower* 71
How to design a good game 9

*Inbetween* 83
Instructions 6

Keyboard 1
**KEYBOARD GUIDE** 122

Logical values 38,85

Method 11
*Money Match* 20

**NUMBER GAMES** 17

OVER 25

*Pace the Plank* 35
*Playing the Numbers* 80
*Pursuit* 112

Random numbers 17,108
*Recognize the Enemy* 25

SCREEN$ 29
Setting up the TS2068 1
*Shrink* 38
Shuffling 34,93
**SIMULATION GAMES** 59
*Sink the Ship* 59
*Solitaire* 54
**STARTING ON TS2068 GAMES** 1
Suits 93

Testing 15
*Treasure Trove* 105
*Triple* 76
TV tuning 3
Using This Book 5
Validations 13,58,75,85,118

**WORD GAMES** 29

# KEYBOARD GUIDE

## MODES

K automatic
L automatic
C CAPS LOCK (CAPS SHIFT AND 2)
E CAPS SHIFT AND SYMBOL SHIFT
G GRAPHICS (CAPS SHIFT AND 9)

| ENTRY | MODE | SHIFT | KEY |
|---|---|---|---|
| a | L | | A |
| A | L | CAPS | A |
| ABS | E | | G |
| ACS | E | CAPS or SYMBOL | W |
| AND | K, L or C | SYMBOL | Y |
| ASN | E | CAPS OR SYMBOL | Q |
| AT | K, L or C | SYMBOL | I |
| ATN | E | CAPS or SYMBOL | E |
| ATTR | E | CAPS or SYMBOL | L |
| b | L | | B |
| B | L | CAPS | B |
| BEEP | E | CAPS or SYMBOL | Z |
| BIN | E | | B |
| BORDER | K | | B |
| BRIGHT | E | CAPS or SYMBOL | B |
| c | L | | C |
| C | L | CAPS | C |
| CHR$ | E | | u |
| CIRCLE | E | CAPS or SYMBOL | |
| CLEAR | K | | X |
| CLS | K | | V |
| CODE | E | | I |
| COPY | K | | Z |
| COS | E | | W |
| d | L | | D |
| D | L | CAPS | D |
| DATA | E | | D |
| DEF FN | E | SYMBOL | 1 |
| DIM | K | | D |
| DRAW | K | | W |
| e | L | | E |

| ENTRY | MODE | SHIFT | KEY |
|---|---|---|---|
| E | L | CAPS | E |
| EXP | E | | X |
| f | L | | F |
| F | L | CAPS | F |
| FLASH | E | CAPS or SYMBOL | V |
| FN | E | SYMBOL | 2 |
| FOR | K | | F |
| g | L | | G |
| G | L | CAPS | G |
| GO SUB | K | | H |
| GO TO | K | | G |
| h | L | | H |
| H | L | CAPS | H |
| i | L | | I |
| I | L | CAPS | I |
| IF | K | | U |
| IN | E | CAPS or SYMBOL | I |
| INK | E | CAPS or SYMBOL | X |
| INKEY$ | E | | N |
| INPUT | K | | I |
| INT | E | | R |
| INVERSE | E | CAPS or SYMBOL | M |
| j | L | | J |
| J | L | CAPS | J |
| k | L | | K |
| K | L | CAPS | K |
| l | L | | L |
| L | L | CAPS | L |
| LEN | E | | K |
| LET | K | | L |
| LINE | E | SYMBOL | 3 |
| LIST | K | | K |
| LLIST | E | | V |
| LN | E | | Z |
| LOAD | K | | J |
| LPRINT | E | | C |
| m | L | | M |
| M | L | CAPS | M |
| n | L | | N |
| N | L | CAPS | N |
| NEW | K | | A |
| NEXT | K | | N |

| ENTRY | MODE | SHIFT | KEY |
|---|---|---|---|
| NOT | K, L or C | SYMBOL | S |
| o | L | | O |
| O | L | CAPS | O |
| OR | K, L or C | SYMBOL | U |
| OUT | E | CAPS or SYMBOL | O |
| OVER | E | CAPS or SYMBOL | N |
| p | L | | P |
| P | L | CAPS | P |
| PAPER | E | CAPS or SYMBOL | C |
| PAUSE | K | | M |
| PEEK | E | | O |
| PI | E | | M |
| PLOT | K | | Q |
| POINT | E | SYMBOL | 8 |
| POKE | K | | O |
| PRINT | K | | P |
| q | L | | Q |
| Q | L | CAPS | Q |
| r | L | | R |
| R | L | CAPS | R |
| RANDOMIZE | K | | T |
| READ | E | | A |
| REM | K | | E |
| RESTORE | K | | Y |
| RETURN | K | | Y |
| RND | E | | T |
| RUN | K | | R |
| s | L | | S |
| S | L | CAPS | S |
| SAVE | K | | S |
| SCREEN$ | E | CAPS or SYMBOL | K |
| SGN | E | | F |
| SIN | E | | Q |
| SQR | E | | H |
| STEP | K, L or C | SYMBOL | D |
| STOP | K, L or C | SYMBOL | A |
| STR$ | E | | Y |
| t | L | | T |
| T | L | CAPS | T |
| TAB | E | | P |
| TAN | E | | E |
| THEN | K, L or C | SYMBOL | G |

| ENTRY | MODE | SHIFT | KEY |
|---|---|---|---|
| TO | K, L or C | SYMBOL | F |
| u | L | | U |
| U | L | CAPS | U |
| USR | E | | L |
| v | L | | V |
| V | L | CAPS | V |
| VAL | E | | J |
| VAL$ | E | CAPS or SYMBOL | J |
| VERIFY | E | CAPS or SYMBOL | R |
| w | L | | W |
| W | L | CAPS | W |
| x | L | | X |
| X | L | CAPS | X |
| y | L | | Y |
| Y | L | CAPS | Y |
| z | L | | Z |
| Z | L | CAPS | Z |
| 0 | K, L or C | | 0 |
| 1 | K, L or C | | 1 |
| 2 | K, L or C | | 2 |
| 3 | K, L or C | | 3 |
| 4 | K, L or C | | 4 |
| 5 | K, L or C | | 5 |
| 6 | K, L or C | | 6 |
| 7 | K, L or C | | 7 |
| 8 | K, L or C | | 8 |
| 9 | K, L or C | | 9 |
| ! | K, L or C | SYMBOL | 1 |
| " | K, L or C | SYMBOL | P |
| # | K, L or C | SYMBOL | 3 |
| $ | K, L or C | SYMBOL | 4 |
| % | K, L or C | SYMBOL | 5 |
| & | K, L or C | SYMBOL | 6 |
| ' | K, L or C | SYMBOL | 7 |
| ( | K, L or C | SYMBOL | 8 |
| ) | K, L or C | SYMBOL | 9 |
| * | K, L or C | SYMBOL | B |
| + | K, L or C | SYMBOL | K |
| , | K, L or C | SYMBOL | N |
| — | K, L or C | SYMBOL | J |
| . | K, L or C | SYMBOL | M |
| / | K, L or C | SYMBOL | V |

| ENTRY | MODE | SHIFT | KEY |
|---|---|---|---|
| : | K, L or C | SYMBOL | Z |
| ; | K, L or C | SYMBOL | O |
| < | K, L or C | SYMBOL | R |
| = | K, L or C | SYMBOL | L |
| > | K, L or C | SYMBOL | T |
| ? | K, L or C | SYMBOL | C |
| @ | K, L or C | SYMBOL | 2 |
| [ | E | CAPS or SYMBOL | Y |
| \ | E | CAPS or SYMBOL | D |
| ] | E | CAPS or SYMBOL | U |
| ↑ | K, L or C | SYMBOL | H |
| _ | K, L or C | SYMBOL | 0 |
| £ | K, L or C | SYMBOL | X |
| { | E | CAPS or SYMBOL | F |
| \| | E | CAPS or SYMBOL | S |
| } | E | CAPS or SYMBOL | G |
| ~ | E | CAPS or SYMBOL | A |
| © | E | CAPS or SYMBOL | P |
| < = | K, L or C | SYMBOL | Q |
| > = | K, L or C | SYMBOL | E |
| < > | K, L or C | SYMBOL | W |
| ▙ | G | | 1 |
| ▟ | G | | 2 |
| ▄ | G | | 3 |
| ▛ | G | | 4 |
| ▌ | G | | 5 |
| ▞ | G | | 6 |
| ▚ | G | | 7 |
| █ | G | | 8 |
| ★ | K, L or C | | SPACE |

Computers/Timex Sinclair 2068 $7.95

# CREATIVE GAMES

## FOR THE TIMEX SINCLAIR 2068

**ROBERT MAUNDER**

This book contains twenty-one colorful and entertaining number games, word games, simulations, and games of chance for the Timex Sinclair 2068 computer, including:

- CRACK THE COMBINATION
- AWFUL ANAGRAMS
- SHRINK
- SOLITAIRE
- DEAD DUCKS
- GRAB THE GOBLIN
- BLACKJACK
- SINK THE SHIP
- TREASURE TROVE

Each program listing in this book is accompanied by clear instructions for entry and play, and the author includes suggestions on how to modify the programs. Besides providing hours of entertainment, these games can serve as a fascinating introduction to programming in BASIC on the Timex Sinclair 2068.

CREATIVE COMPUTING PRESS

ISBN: 0-916688-24-0